GENESIS User Guide

1.6.0

RIKEN

Dec 21, 2020

GENESIS 1.6.0

Project Leader: Yuji Sugita (RIKEN)

Current main developers: Jaewoon Jung (RIKEN), Chigusa Kobayashi (RIKEN), Takaharu Mori (RIKEN), Hiraku Oshima (RIKEN), Kiyoshi Yagi (RIKEN)

Current developers/contributors: Shingo Ito (RIKEN), Motoshi Kamiya (RIKEN/IMS), Kento Kasahara (RIKEN/Osaka Univ.), Yasuhiro Matsunaga (RIKEN/Saitama Univ.), Daisuke Matsuoka (RIKEN/RIST), Osamu Miyashita (RIKEN), Suyong Re (RIKEN/NIBIOHN), Ai Shinobu (RIKEN), Yosuke Sumiya (RIKEN), Florence Tama (RIKEN/Nagoya Univ.), Chen Tan (RIKEN), Isseki Yu (RIKEN/Maebashi Institute of Technology)

Other developers/contributors for older versions: Tadashi Ando (RIKEN), Michael Feig (Michigan State University), Raimondas Galvelis (RIKEN), Ryuhei Harada (RIKEN), Takashi Imai (RIKEN), Yasuaki Komuro (RIKEN), Yasuhito Karino (RIKEN), Naoyuki Miyashita (RIKEN), Wataru Nishima (RIKEN), Donatas Surblys (RIKEN), Koichi Tamura (RIKEN), Kenta Yamada (RIKEN), Takao Yoda (Nagahama Institute of Bio-Science and Technology),

Acknowledgments: Norio Takase (Isogo Soft), Yasumasa Joti (RIKEN SPring8), Akira Naruse (NVIDIA), Yukihiko Hirano (NVIDIA Japan), Hikaru Inoue (Fujitsu Ltd.), Tomoyuki Noda (Fujitsu Ltd.), Kiyotaka Sakamoto (Fujitsu Ltd.), Yoshinobu Akinaga (VINAS), Yoshitake Sakae (RIST)

GENESIS website

https://www.r-ccs.riken.jp/labs/cbrt/

Citation Information

- C. Kobayashi, J. Jung, Y. Matsunaga, T. Mori, T. Ando, K. Tamura, M. Kamiya, and Y. Sugita, "GENESIS 1.1: A hybrid-parallel molecular dynamics simulator with enhanced sampling algorithms on multiple computational platforms", J. Comput. Chem. 38, 2193-2206 (2017).
- J. Jung, T. Mori, C. Kobayashi, Y. Matsunaga, T. Yoda, M. Feig, and Y. Sugita, "GENESIS: A hybrid-parallel and multi-scale molecular dynamics simulator with enhanced sampling algorithms for biomolecular and cellular simulations", WIREs Computational Molecular Science 5, 310-323 (2015).

Copyright Notices

It should be mentioned this package contains the following softwares for convenience. Please note that these are not covered by the license under which a copy of GENESIS is licensed to you, while neither composition nor distribution of any derivative work of GENESIS with these software violates the terms of each license, provided that it meets every condition of the respective licenses.

SIMD-oriented Fast Mersenne Twister (SFMT)

SFMT is a new variant of Mersenne Twister (MT) introduced by Mutsuo Saito and Makoto Matsumoto in 2006. The algorithm was reported at MCQMC 2006. The routine is distributed under the New BSD License.

FFTE: A Fast Fourier Transform Package

FFTE (http://www.ffte.jp/) is written by Daisuke Takahashi (Tsukuba University).

Complementary error function: erfc04

A Complementary error function routine (erfc04) is written by SunSoft, a Sun Microsystems, Inc. business.

L-BFGS-B (version 3.0)

L-BFGS-B (http://users.iems.northwestern.edu/~nocedal/lbfgsb.html) is written by C. Zhu, R. Byrd, J. Nocedal and J. L. Morales.

R. H. Byrd, P. Lu and J. Nocedal. A Limited Memory Algorithm for Bound Constrained Optimization, (1995), SIAM Journal on Scientific and Statistical Computing, 16, 5, pp. 1190-1208.

C. Zhu, R. H. Byrd and J. Nocedal. L-BFGS-B: Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization (1997), ACM Transactions on Mathematical Software, Vol 23, Num. 4, pp. 550-560.

J.L. Morales and J. Nocedal. L-BFGS-B: Remark on Algorithm 778: L-BFGS-B, FORTRAN routines for large scale bound constrained optimization (2011), ACM Transactions on Mathematical Software, Vol 38, Num. 1, Article No. 7.

CONTENTS

CHAPTER

ONE

INTRODUCTION

GENESIS (*Gen*eralized-*E*nsemble *Si*mulation *S*ystem) is a suite of computer programs for carrying out molecular dynamics (MD) simulations of biomolecular systems. MD simulations of biomolecules such as proteins, nucleic acids, lipid bilayers, N-glycans, are used as important research tools in structural and molecular biology. Many useful MD simulation packages [1] [2] [3] [4] [5] are now available together with accurate molecular force field parameter sets [6] [7] [8] [9] [10]. Most of the MD software have been optimized and parallelized for distributed-memory parallel supercomputers or PC-clusters. Therefore hundreds of CPUs or CPU cores can be used efficiently for a single MD simulation of a relatively large biomolecular system, typically composed of several hundred thousands of atoms. In recent years, the number of available CPUs or CPU cores is rapidly increasing. The implmentation of highly efficient parallel schemes is therefore required in modern MD simulation programs. Accelerators such as GPGPU (General-Purpose computing on Graphics Processing Units) also become popular, and thus their utilization is also desired. Actually, many MD program packages support various accelerators.

Our major motivation is to develop MD simulation software with a scalable performance on such modern supercomputers. For this purpose, we have developed the software from scratch, introducing the hybrid (MPI + OpenMP) parallelism, several new parallel algorithms [11] [12], and GPGPU calculation. Another motivation is to develop a simple MD program, which can be easily understood and modified for methodological developments. These two policies (high parallel performance and simplicity) usually conflict each other in computer software. To avoid the conflict, we have developed two MD programs in **GENESIS**, namely **SPDYN** (*Sp*atial decomposition *dyn*amics) and **ATDYN** (*At*omic decomposition *dyn*amics).

SPDYN and **ATDYN** share almost the same data structures, subroutines, and modules, but differ in their parallelization schemes. In **SPDYN**, the spatial decomposition scheme is implemented with new parallel algorithms [11] [12] and GPGPU calculation. In **ATDYN**, the atomic decomposition scheme is introduced for simplicity. The performance of **ATDYN** is not comparable to **SPDYN** due to the simple parallelization scheme. However, **ATDYN** is easier to modify for development of new algorithms or novel molecular models. We hope that users develop new methodologies in **ATDYN** at first and, eventually, port them to **SPDYN** for the better performance. As we maintain consistency between the source codes of **ATDYN** and **SPDYN**, switching from **ATDYN** to **SPDYN** is not quite hard.

Other features in **GENESIS** are listed below:

- Not only atomistic molecular force field (CHARMM, AMBER) but also some coarse-grained models are available in **ATDYN**.
- For extremely large biomolecular systems (more than 10 million atoms), parallel input/output (I/O) scheme is implemented.
- **GENESIS** is optimized for K and Fugaku computer (developed by RIKEN and Fujitsu company), but it is also available on Intel-based supercomputers and PC-clusters.

- **GENESIS** is written in modern Fortran language (90/95/2003) using modules and dynamic memory allocation. No common blocks are used.
- **GENESIS** is free software under the GNU Lesser General Public License (LGPL) version 3 or later. We allow users to use/modify **GENESIS** and redistribute the modified version under the same license.

This user manual mainly provides detailed description of keywords used in the control file. Tutorials for standard MD simulations, REMD simulations, and some analyses are available online (https://www.r-ccs.riken.jp/labs/cbrt/). We recommend new users of **GENESIS** to start from the next chapter to learn a basic idea, installation, and work flow of the program.

Comparing to other MD software, e.g. AMBER, CHARMM, or NAMD, **GENESIS** is a very young MD simulation program. Before releasing the program, the developers and contributors in **GENESIS** development team worked hard to fix all bugs in the program, and performed a bunch of test simulations. Still, there might be defects or bugs in **GENESIS**. Since we cannot bear any responsibility for the simulation results produced by **GENESIS**, we strongly recommend the users to check the results carefully.

The **GENESIS** development team has a rich plan for future development of methodology and molecular models. We would like to make **GENESIS** one of the most powerful and feasible MD software packages, contributing to computational chemistry and biophysics. Computational studies in life science is still at a very early stage (like 'GENESIS') compared to established experimental researches. We hope that **GENESIS** pushes forward the computational science and contribute to bio-tech and medical applications in the future.

CHAPTER

TWO

GETTING STARTED

2.1 Installation of GENESIS

2.1.1 Requirements

Compilers

GENESIS works on various systems: laptop PCs, workstations, cluster machines, and supercomputers. Since the source code of **GENESIS** is mainly written in Fortran language, Fortran compiler is the first requirement for installation. In addition, "preprocessor" is required, because the source code is "processed" according to the user's computer environment before the compilation. One of the commonly used Fortran compilers is `gfortran`, which is freely available as part of the GNU Compiler Collection (GCC). In this case, `cpp` is selected as a preprocessor, which is also available freely. Another recommended Fortran compiler is `ifort` provided by Intel Corporation which enables us to run the program much faster on Intel CPU. In the Intel compiler package, `fpp` is provided as a preprocessor. Fujitsu compiler `frtpx`, which also functions as a preprocessor, is suitable for Fujitsu machines like FX100.

MPI and OpenMP

Both **ATDYN** and **SPDYN** work on multiple CPU cores using MPI (Message Passing Interface) and OpenMP protocols (hybrid MPI+OpenMP). MPI and OpenMP are commonly used for parallel computing. In general, MPI is employed for communication between different machines, nodes, or processors, where the memory is not shared among them (distributed-memory). On the other hand, OpenMP is employed in a single processor, and thus, memory is shared in the parallel computation.

OpenMP is natively supported in most modern Fortran compilers. As for MPI, however, the users may have to install MPI libraries by themselves, especially, in the case of laptop PCs and workstations. One of the commonly used MPI software is OpenMPI (https://www.open-mpi.org/). When the users install the OpenMPI libraries in the computer, the users must specify Fortran and C compilers (e.g., `gfortran` and `gcc`) to be used with MPI. After installing the libraries, the users can use `mpif90`, `mpicc`, and `mpirun`, which are necessary to compile and run the program that is parallelized with MPI. OpenMPI is available freely, and the example installation scheme is shown in *Appendix*. Intel and Fujitsu Corporations are also providing their own MPI libraries for parallel computation.

Mathematical libraries

GENESIS utilizes mathematical libraries such as LAPACK and BLAS (http://www.netlib.org/lapack/). These libraries enable us to efficiently solve complicated mathematical equations such as eigenvalue problems and singular value decomposition. The users have to install these libraries by themselves, if they are not installed in the computer (see *Appendix*). In the case of the Intel and Fujitsu compilers, Intel MKL and Fujitsu SSL II are automatically selected, respectively.

GPGPU

SPDYN works not only with CPU but also with CPU+GPU. Some of the source code in **SPDYN** are written in CUDA, which enables us to effectively run the program on NVIDIA GPU cards. If the users want to run **SPDYN** with GPGPU calculations, the CUDA toolkit (https://developer.nvidia.com/cuda-toolkit) must be also installed in the computer. Note that OpenACC is not employed in **GENESIS** currently.

The recommended compilers, preprocessors, and libraries for **GENESIS** are listed below. Please make sure that at least one of them in each section is installed on your system (GPU is optional). If the users do not use the Intel or Fujitsu compilers, the combination of GCC compiler, GCC preprocessor, and OpenMPI is recommended.

- Operating systems (see *Appendix*)
  - Linux
  - macOS
- Fortran and C compilers
  - GCC compiler `gfortran`, `gcc` (version 4.4.7 or higher is required)
  - Intel compiler `ifort`, `icc`
  - Fujitsu compiler `frtpx`, `fccpx`
- Preprocessors
  - GCC preprocessor `cpp`
  - Intel preprocessor `fpp`
  - Fujitsu compiler `frtpx`
- MPI libraries for parallel computing
  - OpenMPI `mpirun`, `mpif90`, `mpicc`
  - Intel MPI
  - Fujitsu MPI
- Numerical libraries for mathematical algorithms
  - LAPACK/BLAS
  - Intel Math Kernel Library (MKL)
  - Fujitsu Scientific Subroutine Library (SSL II)
- GPU **(Optional)**
  - NVIDIA GPU cards which support Compute Capability (CC) 3.5 or higher
  - The following GPU cards and CUDA versions have been tested by the GENESIS developers
    - NVIDIA K20, K40, P100, TITAN V, GTX 1080, GTX 1080Ti, RTX 2080, RTX 2080Ti

      * CUDA ver. 8.0, 9.0, 9.1, 9.2, 10.0

Note: If you are using a supercomputer in universities or research institutes, there is a high chance that the system already provides the above requirements so that you don't need to install yourself. Please refer to the users' guide of the supercomputer, or consult the system administrator.

In general, the latest version of CUDA does not support the latest version of GCC compiler. If you cannot compile GENESIS with new CUDA (ver. 10) and new GCC compiler (ver. 8.0 or higher), please first make an attempt to install CUDA with older GCC compilers (ver. 7.0 or older), and then install GENESIS with those CUDA and GCC compilers.

2.1.2 General scheme for installation

Step1. Download the source code

The source code of **GENESIS** is available in the GENESIS website (https://www.r-ccs.riken.jp/labs/cbrt/download/). The users have to first uncompress the download file in an appropriate directory. Here, we assume that the users install **GENESIS** in "$HOME/genesis". The "src" directory contains the source code, and "COPYING" is the software license.

```
$ mkdir $HOME/genesis
$ cd $HOME/genesis
$ mv ~/Downloads/genesis-1.6.0.tar.bz2 ./
$ tar xvfj genesis-1.6.0.tar.bz2
$ cd genesis-1.6.0
$ ls
AUTHORS       Makefile.am    aclocal.m4      depcomp       src
COPYING       Makefile.in    compile         fortdep.py
ChangeLog     NEWS           configure       install-sh
INSTALL       README         configure.ac    missing
```

Step2. Configure

In order to compile the source code, the users execute the "configure" script in the directory. This script automatically detects appropriate compilers, preprocessors, and libraries in the users' computer, and create "`Makefile`".

```
$ ./configure
```

If you encountered a failure in the configure command, please check the error message carefully. You may have to add appropriate options in this command according to your computer environment (see *Advanced installation*). The followings are possible suggestions to solve frequent problems. Other solutions might be found in the online page (https://www.r-ccs.riken.jp/labs/cbrt/installation/).

- First of all, please check whether the Fortran and C compilers are installed in your computer. If you are going to run GENESIS with multiple CPUs, you should additionally install MPI libraries such as OpenMPI before compiling GENESIS (see *Appendix*).
- If you see the error message "configure: error: Fortran compiler cannot create executables", it may imply that the path to the installed compilers or MPI libraries might not be correctly set in "~/.bashrc" or "~/.bash_profile" (see *Appendix*). This configure script automatically detects "mpif90", "mpifrtpx", or "mpifrt" for Fortran compiler, and "mpicc", "mpifccpx", or "mpifrt" for C compiler. The error message may indicate that the detection was failed due to some reasons. For example, if you installed OpenMPI in your computer, both "mpif90" and "mpicc" should be detected. Please check the path to these executables by typing the "which" command (e.g., which mpif90) in the terminal window. If you cannot see any paths, setting of the path in "~/.bashrc" or "~/.bash_profile" might have a mistake (see *Appendix*). You should also check typing mistakes of the path.
- If the recommended software are not used in compilation, warning messages might be displayed in the terminal when the configure command is executed. Those messages are just a warning (not an error), and you may continue the compilation. However, we strongly recommended you to verity the installation in such cases (see *Verify the installation*).

- In some supercomputer systems, “module load [module]” command is required to use compliers, and need to be set before the configure. See the user guide of the system.
- Try “autoreconf” or “./bootstrap” before the configure command, if your computer environment is significantly different from what we assume and/or if you modify “configure.ac” or “Makefile.am” by yourself.

Step3. Make install

After the “configure” command is successful, type the following command to compile and install **GENESIS**. All programs in **GENESIS** are compiled and installed into the “./bin” directory by default.

```
$ make install
```

If you encountered a failure, please check the error message carefully. In many cases, errors are caused by invalid path of compilers and libraries. The followings are possible suggestions to solve frequent problems. Other solutions might be found in the online page (https://www.r-ccs.riken.jp/labs/cbrt/installation/).

- If the error message is like “/usr/bin/ld: cannot find -lblas” or “/usr/bin/ld: cannot find -llapack”, make sure that the BLAS or LAPACK libraries are installed in the computer (see also *Appendix*). The users may also have to set the path to the libraries in the “configure” command with the “`LAPACK_LIBS`” or “`LAPACK_PATH`” option (see *Advanced installation*).
- If you have installed additional software or libraries to solve a make error, please execute “make clean”, and try Step2 and “make install” again.

Step4. Confirmation

After the installation is successfully finished, the following binary files are found in the “bin” directory. There are 42 programs in total. Brief description of each program is shown in *Available Programs*.

```
$ ls ./bin
atdyn                 fret_analysis        qmmm_generator
avecrd_analysis       hb_analysis          qval_analysis
comcrd_analysis       hbond_analysis       rdf_analysis
contact_analysis      kmeans_clustering    remd_convert
crd_convert           lipidthick_analysis  rg_analysis
density_analysis      mbar_analysis        rmsd_analysis
diffusion_analysis    meanforce_analysis   rpath_generator
distmat_analysis      msd_analysis         rst_convert
drms_analysis         pathcv_analysis      rst_upgrade
dssp_interface        pcavec_drawer        sasa_analysis
eigmat_analysis       pcrd_convert         spdyn
emmap_generator       pmf_analysis         tilt_analysis
energy_analysis       prjcrd_analysis      trj_analysis
flccrd_analysis       prst_setup           wham_analysis
```

2.1.3 Advanced installation

In the above scheme, **GENESIS** is installed with default options, and all installed programs run on CPU with double precision calculation. The users can specify additional options in the configure command according to the users' computer environment or desired conditions. The full lists of the available options are obtained by "`./configure --help`". The representative options are as follows.

`--enable-single`

Turn on single precision calculation. In this case, only **SPDYN** is installed.

`--enable-gpu`

Turn on GPGPU calculation. In this case, only **SPDYN** is installed.

`--with-cuda=PATH`

Define path to the CUDA libraries manually.

`--disable-mpi`

Turn off MPI parallelization. In this case, **SPDYN** is not installed.

`--disable-openmp`

Turn off OpenMP parallelization.

`--disable-parallel_IO`

Do not install the parallel I/O tool (**prst_setup**)

`--enable-debug`

Turn on program debugging (see below)

`--prefix=PREFIX`

Install the programs in the directory designated by `PREFIX`

Configuration with non-default compilers

Although the compilers are set to "mpif90" and "mpicc" by default, the users may specify different compilers by configure commands. Fortran compiler is specified with `FC` and `F77`, and C compiler with `CC`. For example, in the case of "mpiifort" and "mpiicc", the following options are added:

```
$ ./configure CC=mpiicc FC=mpiifort F77=mpiifort
```

Configuration with an explicit path to LAPACK/BLAS libraries

The following is an example command to set the path to LAPACK and BLAS libraries that are installed in $HOME/Software/lapack-3.8.0/ (see also *Appendix*). Please be careful about the filename of the installed libraries. If the BLAS libraries are installed as "librefblas.a", the option "-lrefblas" must be used. If "librefblas.a" is renamed to "libblas.a", the following command can be used. Linking with the reverse order of "-llapack" and "-lblas" might also cause a failure of installation of **GENESIS**.

```
$ ./configure LAPACK_LIBS="-L$HOME/Software/lapack-3.8.0 -llapack -lblas"
```

or use the “LAPACK_PATH” option:

```
$ ./configure LAPACK_PATH=$HOME/Software/lapack-3.8.0
```

Configuration for single-precision calculation on CPU

The following command is used to turn on single-precision calculation in **SPDYN**. In this case, force calculations are carried out with single precision, while integration of the equations of motion as well as accumulation of the force and energy are still done with double-precision.

```
$ ./configure --enable-single
```

Only **SPDYN** that works on CPU will be installed with this option. If the user additionally needs analysis tools as well as **ATDYN**, one must prepare another GENESIS directory, and install without the “--enable-single” option.

Configuration for GPGPU calculation

In the following command, the users install **SPDYN** that works on CPU+GPU with single-precision calculation. If “--enable-single” is omitted in the command, **SPDYN** works on CPU+GPU with double-precision calculation.

```
$ ./configure --enable-single --enable-gpu
```

Here, if the users encountered an error message like “nvcc: command not found”, make sure that the CUDA Toolkit is installed in the computer. In typical Linux workstations or cluster machines, CUDA is installed in “/usr/local/cuda-x.x/” or “/usr/lib/x86_64-linux-gnu/”, and “nvcc” should be in a “bin” directory of the install directory. The path to “nvcc” and CUDA libraries should be set in a startup file such as “~/.bashrc”. For example, add the following information to “~/.bashrc” in the case of CUDA 9.0,

```
CUDAROOT=/usr/local/cuda-9.0
export PATH=$CUDAROOT/bin:$PATH
export LD_LIBRARY_PATH=$CUDAROOT/lib64:/lib:$LD_LIBRARY_PATH
```

then reload “~/.bashrc” and try the configure command again. If there still remain some troubles, explicitly specify a path to CUDA libraries in the configure command by:

```
$ ./configure --enable-single --enable-gpu --with-cuda=/usr/local/cuda-9.0
```

Configuration for supercomputer systems

The configuration for supercomputer systems may require non-standard setups. In the online usage page, we describe recommended configure options for some supercomputers (https://www.r-ccs.riken.jp/labs/cbrt/usage/).

For example, the following commands are used to compile **GENESIS** on HOKUSAI GreatWave (FX100) in RIKEN. Note that the parallel I/O tool (**prst_setup**) is not compiled in this configuration, because Fujitsu compiler has a trouble in compiling **prst_setup** (see aldo *Available Programs*).

```
$ module load sparc
$ ./configure --host=k
```

Configuration for single CPU calculations

By specifying the "`--disable-mpi`" option, the users can install **GENESIS** that can work on one CPU. The configure script automatically looks for "gfortran", "ifort", "frt", or "frtpx" for Fortran compiler, and "gcc", "icc", "fcc", or "fccpx" for C compiler. Therefore, in this case MPI libraries are not required for the installation and execution of **GENESIS**. **ATDYN** and analysis tools are installed.

```
$ ./configure --disable-mpi
```

Configuration for program debugging

If the users encountered memory leak errors during the simulation using **GENESIS**, the origin of the error might be tracked by using a program compiled with a debug option. Note that the debug option makes the calculation much slow. In this case, the runtime check is activated only for CPU codes, even if the "–enable-gpu" option is added to the command.

```
$ ./configure --enable-debug=3
```

Note that `--enable-debug` corresponds to `--enable-debug=1`.

- 0 = no debugging (default)
- 1 = debugging without intensive optimization
- 2 = LEVEL1 + debug information (`-g` and `-DDEBUG`)
- 3 = LEVEL2 + memory check (if possible)
- 4 = LEVEL3 + full check (intel compiler only)

Installation using multiple CPU cores (parallel compile)

In Step3, `-j` option is available, which enables quick compilation of the program using multiple CPU cores. The following command uses 4 CPU cores.

```
$ make -j4 install
```

If you encountered an error message like "Fatal Error: Can't delete temporary module file '. . . ': No such file or directory", please try "make install" without the "`-j`" option.

2.1.4 Verify the installation

The users can verify the installation of **GENESIS** by using test sets which are available in the **GENESIS** website (https://www.r-ccs.riken.jp/labs/cbrt/download/). Please, uncompress the download file in an appropriate directory, and move to the "regression_test" directory.

```
$ cd $HOME/genesis
$ mv ~/Downloads/tests-1.6.0.tar.bz2 ./
$ tar xvfj tests-1.6.0.tar.bz2
$ cd tests-1.6.0/regression_test
$ ls
build        test_analysis    test_gamd_spdyn    test_rpath_atdyn
charmm.py    test_atdyn       test_nonstrict.py  test_rpath_spdyn
cleanup.sh   test_common      test_parallel_IO   test_spana
fep.py       test_fep         test_remd.py       test_spdyn
genesis.py   test_fep.py      test_remd_common   test_vib
param        test_gamd.py     test_remd_spdyn    test_vib.py
test.py      test_gamd_atdyn  test_rpath.py
```

In the sub-directories in "regression_test", the users can find a lot of input files ("`inp`"), in which various combinations of simulation parameters are specified. In addition, each sub-directory contains output file ("`ref`") obtained by the developers. The users run "test.py", "test_remd.py", "test_rpath.py", and so on, which enable automatic comparison between the users' and developers' results for each MD algorithm.

Run the basic tests

The following is an example command to verify the two simulators **atdyn** and **spdyn** for basic MD and energy minimization. Here, the programs are executed using 1 CPU core with the "mpirun" command. The users can increase the number of MPI processors according to the users' computer environment, but only 1, 2, 4, or 8 are allowed in these tests. Other MPI launchers such as "mpiexec" are also available in the command. There are about 50 test sets, and each test should finish in a few seconds.

```
$ export OMP_NUM_THREADS=1
$ ./test.py "mpirun -np 1 ~/genesis/genesis-1.6.0/bin/atdyn"
$ ./test.py "mpirun -np 1 ~/genesis/genesis-1.6.0/bin/spdyn"
```

If any tests cannot run, please check the following points:

- Number of OpenMP threads should be specified before running the tests (one is recommended).
- Original executable file name (e.g., **spdyn** and **atdyn**) must not be changed.
- Python ver. 3 does not work. Please use ver. 2.x, and run "`python2.x ./test.py ...`"
- Regression tests via a queuing system or batch script may not work.

The "test.py" script compares energy in log file between the developer's and user's ones. If the energy differences are less than the tolerance (default = 1.00e-08), "Passed" is displayed for each test. Among the physical quantities in the log file, virial is the most sensitive to numerical factors, and thus, the tolerance for virial is set to a larger value (1.00e-06). After all tests are finished, the total number of succeeded, failed, and aborted runs will be displayed at the end.

```
Passed  46 / 46
Failed  0 / 46
Aborted 0 / 46
```

If all tests were passed, it means that your **GENESIS** can generate identical results to the developer's **GENESIS**. Note that the developer's **GENESIS** was compiled with Intel compilers, Intel MKL, OpenMPI library, and the double precision option on Intel CPUs. If your computer system is significantly different from the developer's one, unexpected numerical errors may happen, which can cause failures in some tests. If there were any aborted tests, the users had better check their log or error files carefully, which exist in the tested sub-directory, and figure out why the error happened. The followings are suggestions to solve typical problems:

- If some tests were aborted due to "memory allocation error", the reason might come from limitation of the memory size. Namely, those tested systems were too large for your computer. The problem should not be so serious.
- Available number of MPI slots in your computer might be actually smaller than the given number of MPI processors. Try to use less number of MPI processors.
- Try to specify the "absolute path" to the program instead of using "relative path".
- Make sure that the MPI environment is properly set.
- Detailed solutions in specific supercomputer systems might be found in the GENESIS website (https://www.r-ccs.riken.jp/labs/cbrt/usage/).

Run the additional tests

By using a similar way, the users can check other functions in **atdyn** and **spdyn**, such as GaMD, REMD, path sampling, vibrational analysis, parallel I/O, and GPGPU calculation. Available number of MPI processors depends on each test (test_gamd: 1, 2, 4, 8; test_remd: 4, 8, 16, 32; test_rpath: 8; test_vib: 8; parallel_io: 8; gpu: 1, 2, 4, 8). As for the GPGPU tests, the users must use **spdyn** that was installed with the "–enable-gpu" option. The parallel_io tests require both **spdyn** and **prst_setup**. Note that **prst_setup** is not installed in some cases according to the configure options or compilers (see *Advanced installation*). In order to run the analysis tool tests, the users first move to "test_analysis", and then execute "./test_analysis.py". Note that MPI is not used in the analysis tool tests. In a similar way, the users can test the SPANA (spatial decomposition analysis) tool sets. SPANA tool sets are tested with 1, 2, 4, and 8 MPI processes.

```
$ export OMP_NUM_THREADS=1
$ ./test_gamd.py "mpirun -np 1 ~/genesis/genesis-1.6.0/bin/atdyn"
$ ./test_gamd.py "mpirun -np 1 ~/genesis/genesis-1.6.0/bin/spdyn"
$ ./test_remd.py "mpirun -np 4 ~/genesis/genesis-1.6.0/bin/atdyn"
$ ./test_remd.py "mpirun -np 4 ~/genesis/genesis-1.6.0/bin/spdyn"
$ ./test_fep.py "mpirun -np 8 ~/genesis/genesis-1.6.0/bin/spdyn"
$ ./test_rpath.py "mpirun -np 8 ~/genesis/genesis-1.6.0/bin/atdyn"
$ ./test_rpath.py "mpirun -np 8 ~/genesis/genesis-1.6.0/bin/spdyn"
$ ./test_vib.py   "mpirun -np 8 ~/genesis/genesis-1.6.0/bin/atdyn"
$ ./test.py "mpirun -np 8 ~/genesis/genesis-1.6.0/bin/spdyn" parallel_io
$ ./test.py "mpirun -np 8 ~/genesis/genesis-1.6.0/bin/spdyn" gpu

$ cd test_analysis
$ ./cleanup.sh
```

(continues on next page)

(continued from previous page)

```
$ export OMP_NUM_THREADS=1
$ ./test_analysis.py ~/genesis/genesis-1.6.0/bin/

$ cd test_spana
$ ./cleanup.sh
$ export OMP_NUM_THREADS=1
$ ./test_spana.py ~/genesis/genesis-1.6.0/bin/
```

Note: Some tests might be using “abnormal” parameters or conditions in the input files for the sake of simple tests. Do not use such parameters in your research. “Normal” parameters are mainly introduced in this user manual or online tutorials.

2.1.5 Clean install and re-compilation

The following commands are used to fully recompile **GENESIS**. Note that the direct "make clean" command may not work in the case where `Makefiles` were created in another machine. In this case, the users must run the "./configure" command before "make clean".

```
$ make clean
$ make distclean
$ ./configure [option]
$ make install
```

2.1.6 Uninstall

The user can uninstall **GENESIS** by just removing the program directory. If the user changed the install directory by specifying "`--prefix=PREFIX`" in the configure command, please remove the programs (**atdyn**, **spdyn**, and so on) in the "`PREFIX`" directory.

```
$ rm -rf $HOME/genesis/genesis-1.6.0
```

2.2 Basic usage of GENESIS

2.2.1 Running GENESIS on a command line

The **GENESIS** programs are executed on a command line. The first argument is basically interpreted as an input file of the program. The input file, which we call *control file* hereafter, contains parameters for simulations. The following examples show typical usage of the **GENESIS** programs. In the case of serial execution,

```
$ [program_name] [control_file]
```

In the case of parallel execution with "mpirun",

```
$ mpirun -np n [program_name] [control_file]
```

For example, **SPDYN** is executed in the following way using 8 MPI processors:

```
$ mpirun -np 8 ~/genesis/genesis-1.6.0/bin/spdyn INP
```

The users should specify an OpenMP thread number explicitly before running the program. Appropriate number of CPU cores must be used according to the user's computer environment (see also *Available Programs*). For example, if the users want to use 32 CPU cores in the calculation, the following command might be executed.

```
$ export OMP_NUM_THREADS=4
$ mpirun -np 8 ~/genesis/genesis-1.6.0/bin/spdyn INP
```

As for the analysis tools, the usage is almost same, but mpirun is not used. Note that some analysis tools (e.g., mbar_analysis, wham_analysis, msd_analysis, and drms_analysis) are parallelized with OpenMP.

```
# RMSD analysis tool
$ ~/genesis/genesis-1.6.0/bin/rmsd_analysis INP

# MBAR analysis
$ export OMP_NUM_THREADS=4
$ ~/genesis/genesis-1.6.0/bin/mbar_analysis INP
```

2.2.2 Automatic generation of a template control file

Basic usage of each program is shown by executing the program with the `-h` option. In addition, sample control file of each program can be obtained with the `-h ctrl` option:

```
# Show the usage of the program
$ [program_name] -h

# Display a template control file
$ [program_name] -h ctrl [module_name]
```

For example, in the case of **SPDYN**, the following messages are displayed:

```
$ spdyn -h

# normal usage
  % mpirun -np XX ./spdyn INP

# check control parameters of md
  % ./spdyn -h ctrl md

# check control parameters of min
  % ./spdyn -h ctrl min

# check control parameters of remd
  % ./spdyn -h ctrl remd

# check control parameters of rpath
  % ./spdyn -h ctrl rpath

# check all control parameters of md
  % ./spdyn -h ctrl_all md

(skipped...)
```

This message tells the users that **SPDYN** can be executed with mpirun. A template control file for molecular dynamics simulation (md) can be generated by executing **SPDYN** with the `-h ctrl md` option. The same way is applicable for energy minimization (min), replica exchange simulation (remd), and replica path sampling simulation (rpath). The template control file for energy minimization is shown below. If the users want to show all available options, please specify `ctrl_all` instead of `ctrl`. The users can edit this template control file to perform the simulation that the users want to do.

```
$ ~/genesis/genesis-1.6.0/bin/spdyn -h ctrl min > INPMIN

$ less INPMIN

[INPUT]
topfile = sample.top      # topology file
parfile = sample.par      # parameter file
psffile = sample.psf      # protein structure file
pdbfile = sample.pdb      # PDB file

[ENERGY]
forcefield    = CHARMM    # [CHARMM,AMBER,GROAMBER,GROMARTINI]
electrostatic = PME       # [CUTOFF,PME]
switchdist    = 10.0      # switch distance
cutoffdist    = 12.0      # cutoff distance
pairlistdist  = 13.5      # pair-list distance

[MINIMIZE]
method        = SD        # [SD]
nsteps        = 100       # number of minimization steps

[BOUNDARY]
type          = PBC       # [PBC, NOBC]
```

2.3 Control file

In the control file, detailed simulation conditions are specified. The control file consists of several sections (e.g., **[INPUT]**, **[OUTPUT]**, **[ENERGY]**, **[ENSEMBLE]**, and so on), each of which contains closely-related keywords. For example, in the **[ENERGY]** section, parameters are specified for the potential energy calculation such as a force field type and cut-off distance. In the **[ENSEMBLE]** section, there are parameters to select the algorithm to control the temperature and pressure in addition to the target temperature and pressure of the system. Here, we show example control files for the energy minimization and normal molecular dynamics simulations.

2.3.1 Example control file for the energy minimization

The control file for the energy minimization must include a **[MINIMIZE]** section (see *Minimize section*). By using the following control file, the users carry out 2,000-step energy minimization with the steepest descent algorithm (SD). The CHARMM36m force field is used, and the particle mesh Ewald (PME) method is employed for the calculation of long-range interaction.

```
[INPUT]
topfile = top_all36_prot.rtf      # topology file
parfile = par_all36m_prot.prm     # parameter file
strfile = toppar_water_ions.str   # stream file
psffile = build.psf               # protein structure file
pdbfile = build.pdb               # PDB file

[OUTPUT]
dcdfile = min.dcd                 # coordinates trajectory file
rstfile = min.rst                 # restart file

[ENERGY]
forcefield        = CHARMM        # CHARMM force field
electrostatic     = PME           # Particl mesh Ewald method
switchdist        = 10.0          # switch distance (Ang)
cutoffdist        = 12.0          # cutoff distance (Ang)
pairlistdist      = 13.5          # pair-list cutoff distance (Ang)
pme_nspline       = 4             # order of B-spline in PME
pme_max_spacing   = 1.2           # max grid spacing allowed (Ang)
vdw_force_switch  = YES           # turn on van der Waals force switch
contact_check     = YES           # turn on clash checker

[MINIMIZE]
method            = SD            # Steepest descent method
nsteps            = 2000          # number of steps
eneout_period     =  100          # energy output freq
crdout_period     =  100          # coordinates output frequency
rstout_period     = 2000          # restart output frequency
nbpdate_period    =   10          # pairlist update frequency

[BOUNDARY]
type              = PBC           # periodic boundary condition
box_size_x        = 64.0          # Box size in X dimension (Ang)
box_size_y        = 64.0          # Box size in Y dimension (Ang)
box_size_z        = 64.0          # Box size in Z dimension (Ang)
```

2.3.2 Example control file for normal MD simulations

The control file for normal MD simulations must include a **[DYNAMICS]** section (see *Dynamics section*). By using the following control file, the users carry out a 100-ps MD simulation at T = 298.15 K and P = 1 atm in the NPT ensemble. The equations of motion are integrated by the RESPA algorithm with a time step of 2.5 fs, and the bonds of light atoms (hydrogen atoms) are constrained using the SHAKE/RATTLE and SETTLE algorithms. The temperature and pressure are controlled with the Bussi thermostat and barostat.

```
[INPUT]
topfile = top_all36_prot.rtf     # topology file
parfile = par_all36m_prot.prm    # parameter file
strfile = toppar_water_ions.str  # stream file
psffile = build.psf              # protein structure file
pdbfile = build.pdb              # PDB file
rstfile = min.rst                # restart file

[OUTPUT]
dcdfile = md.dcd                 # coordinates trajectory file
rstfile = md.rst                 # restart file

[ENERGY]
forcefield        = CHARMM       # CHARMM force field
electrostatic     = PME          # Particl mesh Ewald method
switchdist        = 10.0         # switch distance (Ang)
cutoffdist        = 12.0         # cutoff distance (Ang)
pairlistdist      = 13.5         # pair-list cutoff distance (Ang)
pme_nspline       = 4            # order of B-spline in PME
pme_max_spacing   = 1.2          # max grid spacing allowed (Ang)
vdw_force_switch  = YES          # turn on van der Waals force switch

[DYNAMICS]
integrator        =   VRES       # RESPA integrator
timestep          = 0.0025       # timestep (2.5fs)
nsteps            =  40000       # number of MD steps (100ps)
eneout_period     =    400       # energy output period (1ps)
crdout_period     =    400       # coordinates output period (1ps)
rstout_period     =  40000       # restart output period
nbupdate_period   =     10       # nonbond update period
elec_long_period  =      2       # period of reciprocal space calculation
thermostat_period =     10       # period of thermostat update
barostat_period   =     10       # period of barostat update

[CONSTRAINTS]
rigid_bond        = YES          # constraint all bonds involving hydrogen

[ENSEMBLE]
ensemble          = NPT          # NPT ensemble
tpcontrol         = BUSSI        # BUSSI thermostat and barostat
temperature       = 300          # target temperature (K)
pressure          = 1.0          # target pressure (atm)

[BOUNDARY]
type              = PBC          # periodic boundary condition
```

CHAPTER

THREE

AVAILABLE PROGRAMS

3.1 Simulators

3.1.1 Basic functions

atdyn

The simulator that is parallelized with the atomic decomposition scheme. In most cases, **atdyn** is applied to small systems or coarse-grained systems. The program runs on CPU with the hybrid MPI+OpenMP protocol, where only double-precision calculation is available. Since the atomic decomposition is a simple parallelization scheme, the source code is actually simple compared to that for the domain decomposition. Therefore, this program is also useful to develop a new function of **GENESIS**.

spdyn

The simulator that is parallelized with the domain decomposition scheme. The program is designed to achieve high-performance molecular dynamics simulations, such as microsecond simulations and cellular-scale simulations. The program runs on not only CPU but also CPU+GPU with the hybrid MPI+OpenMP protocol. Here, beside double-precision, mixed-precision calculations are also available. In the mixed-precision model, force calculations are carried out with single precision, while integration of the equations of motion as well as accumulation of the force and energy are done with double-precision.

Table 3.1: Available functions in **atdyn** and **spdyn**

Function	atdyn	spdyn
Energy minimization	◯	◯
All-atom molecular dynamics	◯	◯
Coarse-grained molecular dynamics	◯	◯ (All-atom Gō model)
Implicit solvent model	◯	–
Replica-exchange method	◯	◯
Gaussian accelerated MD	◯	◯
String method	◯	◯
QM/MM calculation	◯	–
Vibrational analysis	◯	–
Cryo-EM flexible fitting	◯	◯
Precision	double	double/mixed
GPGPU calculation	–	◯ (All-atom MD)
Parallel I/O	–	◯

3.1.2 Atomic and domain decomposition schemes

In the atomic decomposition MD, which is also called a replicated-data MD algorithm, all MPI processors have the same coordinates data of all atoms in the system. MPI parallelization is mainly applied to the "DO loops" of the bonded and non-bonded interaction pair lists for the energy and force calculations. Fig. 3.1 (a) shows a schematic representation of the atomic decomposition scheme for the non-bonded interaction calculation in a Lennard-Jones system, where 2 MPI processors are used. In this scheme, MPI_ALLREDUCE must be used to accumulate all the atomic forces every step, resulting in large communication cost.

In the domain-decomposition MD, which is also called a distributed-data MD algorithm, the whole system is decomposed into domains according to the number of MPI processors, and each MPI processor is assigned to a specific domain. Each MPI processor handles the coordinates data of the atoms in the assigned domain and in the buffer regions near the domain boundary, and carries out the calculation of the bonded and non-bonded interactions in the assigned domain, enabling us to reduce computational cost. In this scheme, communication of the atomic coordinates and forces in the buffer region is essential. Fig. 3.1 (b) is a schematic representation of the domain decomposition scheme, where the system is decomposed into two domains to use 2 MPI processors. Note that in the figure the system periodicity is not considered for simplicity.

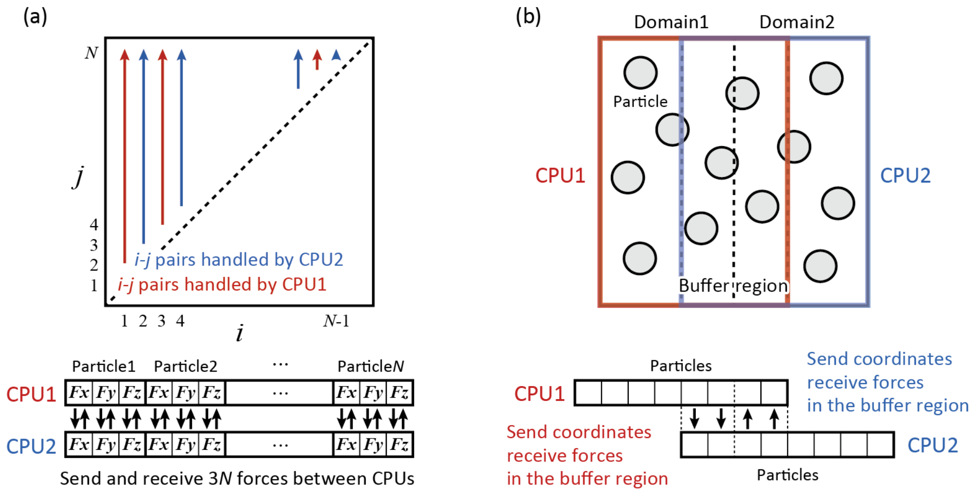


Fig. 3.1: Parallelization scheme in the (a) atomic decomposition and (b) domain decomposition.

3.1.3 Hybrid MPI+OpenMP calculation in SPDYN

The users had better understand a basic scheme of parallel calculation in **SPDYN** to get the best performance in the calculation. As described above, the simulation box is divided into domains according to the number of MPI processors. Each domain is further divided into smaller cells, each of whose size is adjusted to be approximately equal to or larger than the half of "pairlistdist + α". Here, "pairlistdist" is specified in the control file, and α depends on the algorithms used in the simulation (see the next subsection). Note that all domains or cells have the same size with a rectangular or cubic shape. Each MPI processor is assigned to each domain, and data transfer or communication about atomic coordinates and forces is achieved between only neighboring domains. In addition, calculation of bonded and non-bonded interactions in each domain is parallelized based on the OpenMP protocol. These schemes realize hybrid MPI+OpenMP calculation, which is more efficient than flat MPI calculation on recent computers with multiple CPU cores. Because MPI and OpenMP are designed for distributed-memory and shared-memory architectures, respectively, MPI is mainly used for parallelization between nodes and OpenMP is used within one node.

The following figures illustrate how the hybrid MPI+OpenMP calculations are achieved in **SPDYN**. In Fig. 3.2 (a) and 2(b), 8 MPI processors with 4 OpenMP threads (32 CPU cores in total), and 27 MPI processors with 2 OpenMP threads (54 CPU cores in total) are used, respectively. In these Figures, only XY dimensions are shown for simplicity.

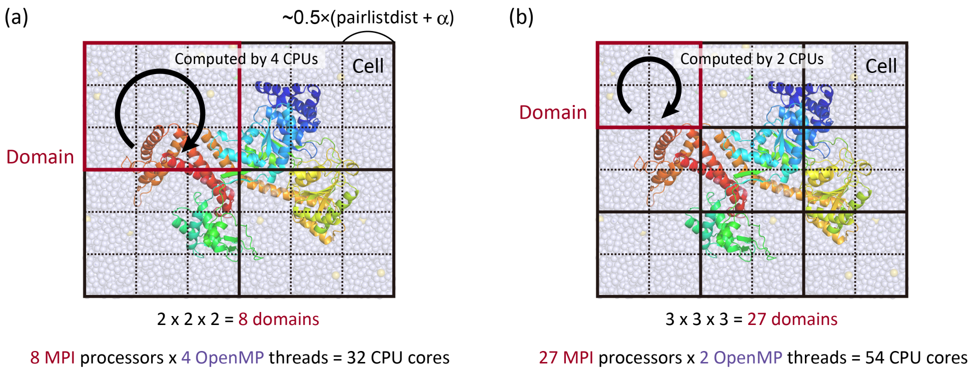


Fig. 3.2: Schematic representation of the hybrid MPI+OpenMP calculation in SPDYN.

For Case (a), the following commands are used:

```
$ export OMP_NUM_THREADS=4
$ mpirun -np 8 ~/genesis/genesis-1.6.0/bin/spdyn INP > log
```

For Case (b), the following commands are used:

```
$ export OMP_NUM_THREADS=2
$ mpirun -np 27 ~/genesis/genesis-1.6.0/bin/spdyn INP > log
```

In the log file, the users can check whether the given numbers of MPI processors and OpenMP threads are actually employed or not. The following information should be found in the log file for instance for Case (a):

```
[STEP2] Setup MPI

Setup_Mpi_Md> Summary of Setup MPI
  number of MPI processes   =          8
  number of OpenMP threads  =          4
  total number of CPU cores =         32
```

Note: In most cases, the number of domains in each dimension is automatically determined according to the given number of MPI processors. However, if such automatic determination is failed, the users must specify the number of domains explicitly in the control file (see *Boundary section*).

3.1.4 Limitation of the available MPI processors

Basically, there is no strict limitation in the available number of MPI processors in **ATDYN**. However, there are a few limitations in **SPDYN**. First, the number of domains must be equal to the number of MPI processors. Second, one domain must be composed of at least 8 cells (= $2\times2\times2$), where the cell size in one dimension is automatically set to be larger than the half of “pairlistdist + α”, The following table summarizes the α value in each algorithm. According to these rules, the available "maximum" number of MPI processors ($N_{\max}$) for a certain target system is mainly determined by the simulation box size and "pairlistdist". For example, if the box size of your target system is $64\times64\times64\text{Å}^3$, and "pairlistdist=13.5" is specified in the control file, $N_{\max}$ is $4\times4\times4 = 64$ in the case of NVT ensemble and "rigid_bond=YES". If the users want to use much more CPU cores than $N_{\max}$, the number of OpenMP threads should be increased instead of the MPI processors.

Rigid bond	Ensemble	α (Å)
NO	NVE/NVT	0.0
NO	NPT/NPAT/NPgT	0.6
YES	NVE/NVT	2.0
YES	NPT/NPAT/NPgT	2.6

In the MD simulation with the NPT ensemble, these rules become more important, because the box size (or cell size) can change during the simulation. In fact, the number of domains in each dimension is initially fixed, but the number of cells can be changed and adjusted to keep the cell size larger than the half of "pairlistdist + α". If the box size is decreased during the simulation, and the number of cells in one dimension of the domain unfortunately becomes one, the calculation stops immediately because of the violation of the above rule. The users may often encounter this situation if the number of cells in one dimension of the domain is just two at the beginning of the MD simulation, and the simulation box has significantly shrunk during the simulation. To avoid such problems, the users may have to use smaller number of MPI processors (which makes cells larger) or shorter pairlistdist (making much cells in one domain), or reconstruct a larger system.

If the users encountered the following error message in the simulation, the problem is probably related to the above rules, where the specified number of MPI processors might exceed $N_{\max}$.

```
Setup_Processor_Number> Cannot define domains and cells. Smaller or
adjusted MPI processors, or shorter pairlistdist, or larger boxsize
should be used.
```

In this case, please make sure that one domain can be composed of at least 8 cells. If the domains and cells are successfully determined, they can be seen in the early part of the log file. The following example is corresponding to the situation in Fig. 3.2 (b).

```
Setup_Boundary_Cell> Set Variables For Boundary Condition
  domains (x,y,z) =          3          3          3
  ncells (x,y,z)  =          6          6          6
```

3.1.5 Available sections

Fundamental functions in **SPDYN** and **ATDYN** are energy minimization (Min), molecular dynamics method (MD), replica-exchange method (REMD), string method (String), and vibrational analysis (Vib). As shown in the last part of the previous chapter, the users carry out simulations of these methods by writing related sections in the control file. The users can extend these fundamental functions by combining various sections. For example, to run a "restrained MD simulation", the users add **[SELECTION]** and **[RESTRAINTS]** sections in the control file of the "normal MD simulation". In fact, there are 17 individual sections in **GENESIS** version 1.4. The following table summarizes the available sections in each function. Detailed usage of each section is described in this user guide, and also in the online tutorials (https://www.r-ccs.riken.jp/labs/cbrt/tutorials2019/).

Table 3.2: Available sections in each algorithm and method

Section	Min	MD	REMD	String	Vib	Description
[INPUT]	○	○	○	○	○	*Input section*
[OUTPUT]	○	○	○	○	○	*Output section*
[ENERGY]	○	○	○	○	○	*Energy section*
[BOUNDARY]	○	○	○	○	○	*Boundary section*
[DYNAMICS]	–	○	○	○	–	*Dynamics section*
[CONSTRAINTS]	–	○	○	○	–	*Constraints section*
[ENSEMBLE]	–	○	○	○	–	*Ensemble section*
[MINIMIZE]	○	–	–	○	○	*Minimize section*
[REMD]	–	–	○	–	–	*REMD section*
[RPATH]	–	–	–	○	–	*RPATH section*
[VIBRATION]	–	–	–	–	○	*Vibration section*
[SELECTION]	○	○	○	○	○	*Selection section*
[RESTRAINTS]	○	○	○	○	○	*Restraints section*
[FITTING]	○	○	○	○	○	*Fitting section*
[GAMD]	–	○	○	–	–	*GAMD section*
[QMMM]	○	○	○	○	○	*QMMM section*
[EXPERIMENTS]	○	○	○	–	–	*Experiments section*

3.2 Analysis tools

The following programs are available as the trajectory analysis tools in **GENESIS**. Basic usage of each tool is similar to that of **spdyn** or **atdyn**. The users can automatically generate a template control file for each program by using the "[program_name] -h ctrl" command. The control file is mainly composed of INPUT, OUTPUT, TRAJECTORY, FITTING, SELECTION, and OPTION sections. The trajectory files to be analyzed are specified in the **[TRAJECTORY]** section, and the parameters used in the analysis are specified in the **[OPTION]** section. Note that the required sections are depending on the program. For example, **eigmat_analysis** requires only INPUT and OUTPUT sections. Detailed usage of each tool is described in the online tutorial.

3.2.1 Trajectory analysis

comcrd_analysis

Analyze the coordinates of the center of mass of the selected atoms.

diffusion_analysis

Analyze the diffusion constant.

distmat_analysis

Analyze the matrix of the averaged distance of the selected atoms.

drms_analysis

Analyze the distance RMSD of the selected atoms with respect to the initial structure.

fret_analysis

Analyze the FRET efficiency.

hb_analysis

Analyze the hydrogen bond.

lipidthick_analysis

Analyze the membrane thickness.

msd_analysis

Analyze the mean-square displacement (MSD) of the selected atoms or molecules.

qval_analysis

Analyze the fraction of native contacts (Q-value).

rg_analysis

Analyze the radius of gyration of the selected atoms.

rmsd_analysis

Analyze the root-mean-square deviation (RMSD) of the selected atoms with respect to the initial structure.

tilt_analysis

Analyze the tilt angle

trj_analysis

Analyze the distance, angle, dihedral angle, distance of the centers of mass (COM) of the selected atom groups, angle of the COM of the selected atom groups, and dihedral angle of the COM of the selected atom groups.

3.2.2 Principal component analysis (PCA)

avecrd_analysis

Calculate the average structure of the target molecule.

flccrd_analysis

Calculate the variance-covariance matrix from the trajectories and averaged coordinates. This tool can be also used to calculate root-mean-square fluctuation (RMSF).

eigmat_analysis

Diagonalize the variance-covariance matrix in PCA.

prjcrd_analysis

Project the trajectories onto PC axes.

pcavec_drawer

Create a script for VMD and PyMol to visualize PC vectors obtained from eigmat_analysis.

3.2.3 Trajectory and restart file converter

crd_convert

Convert trajectories to PDB/DCD formats. This tool can do centering of the target molecule, fitting of a given atom group to the initial structure, wrapping of molecules into the unit cell, combining multiple trajectory files into a single file, extraction of coordinates of selected atoms, and so on.

remd_convert

Convert REMD trajectories to those sorted by parameters. Since the trajectory files are generated from each replica during the REMD simulations, the obtained "raw" trajectories are composed of "mixed" data at various conditions (replica parameters). **remd_convert** enables the users to sort the REMD trajectories by parameters. This is applicable to not only dcdfile but also energy log files.

rst_convert

Convert GENESIS restart file (rstfile) to the PDB file.

rst_upgrade

Convert old restart file (version < 1.1.0) to that in the new format (version >= 1.1.0).

3.2.4 Free energy calculation

wham_analysis

Free energy analysis using the Weighted Histogram Analysis Method (WHAM).

mbar_analysis

Free energy analysis using the Multistate Bennett Acceptance Ratio (MBAR) method.

pmf_analysis

Calculate free energy profile using MBAR output.

meanforce_analysis

Calculate free energy profile from RPATH.

3.2.5 Clustering

kmeans_clustering

Carry out k-means clustering for coordinates trajectories

3.2.6 Interface program

dssp_interface

Interface program to analyze the protein secondary structure in the DCD trajectory file using the DSSP program (https://swift.cmbi.umcn.nl/gv/dssp/).

3.2.7 SPANA

SPANA (SPatial decomposition ANAlysis) is developed to carry out trajectory analyses of large-scale biological simulations using multiple CPU cores in parallel. SPANA employs a spatial decomposition of a system to distribute structural and dynamical analyses into the individual CPU core and allows us to reduce the computational time for the analysis significantly. SPANA is suitable for the analysis of systems with multiple macromolecules (such as cellular crowding systems) under the periodic boundary condition.

contact_analysis

Calculate the number of close atomic pairs between given molecules. The close atomic pairs (or atomic contacts) are defined if the closest atom-atom distance between two macromolecules is shorter than given cutoff distance. This program also finds the closest atom pairs between macromolecule pairs within the cutoff distance.

density_analysis

Calculate 3D density distribution of atoms and output the density in X-PLOR/CCP4/DX format.

hbond_analysis

Analyze hydrogen bonds

rdf_analysis

Calculate the radial distribution function (RDF) and proximal distribution function (PDF) of molecules (as solvent) around the target group (as solute). PDF provides density of solvent as function of the distance to the surface of macromolecules.

sasa_analysis

Calculate solvent accessible surface area (SASA) of the target molecules. This program outputs not only the total SASA but also the SASA for each atom in the target molecules.

3.2.8 Other utilities

rpath_generator

Generate inputs for the string method. This tool is usually used after targeted MD simulation for generating an initial pathway for the subsequent string method.

pathcv_analysis

Calculate tangential and orthogonal coordinates to a pathway from samples.

qmmm_generator

Generate a system for QM/MM calculation from MD data.

emmap_generator

Generate cryo-EM density map from PDB file.

3.3 Parallel I/O tools

SPDYN can be employed with the parallel I/O protocol to achieve massively parallel computation. Since **SPDYN** is parallelized with the domain decomposition scheme, each MPI processor has the coordinates of atoms in the assigned domain. Therefore, large ammount of communication is needed between MPI processors to write the coordinates in a single DCD file, which is a waste of time in the case of the simulations for a huge system like 100,000,000 atoms. To avoid such situations, file I/O in each node (parallel I/O) is useful. The following tools are used to handle the files generated from parallel I/O simulations.

prst_setup

This tool divides input files (PDB and PSF) for a huge system into multiple files, where each file is assinged to each domain. The obtained files can be read as restart files in the **[INPUT]** section. Note that **prst_setup** is not compiled with Fujitsu compilers. Therefore, if the users are going to perform MD simulations with parallel I/O in Fujitsu supercomputers, the users must create the files without using Fujitsu compilers elsewhere in advance. Even if **prst_setup** and **SPDYN** are compiled with different compilers, there is no problem to execute **SPDYN** with parallel I/O.

pcrd_convert

Convert multiple trajectory files obtained from the parallel I/O simulation to a single DCD file. This tool has a similar function to **crd_convert**.

CHAPTER

FOUR

INPUT SECTION

4.1 How to prepare input files

In order to run MD simulations, the users have to prepare input files that contains information about the coordinates of the initial structure as well as topology of the system and force field parameters. The users first create those input files by using a setup tool, and their filenames are specified in the **[INPUT]** section of the control file. **GENESIS** supports various input file formats such as CHARMM, AMBER, and GROMACS. Basically, required input files depend on the force field to be used in the simulation. The following table summarizes the essential input files and setup tools for each force field.

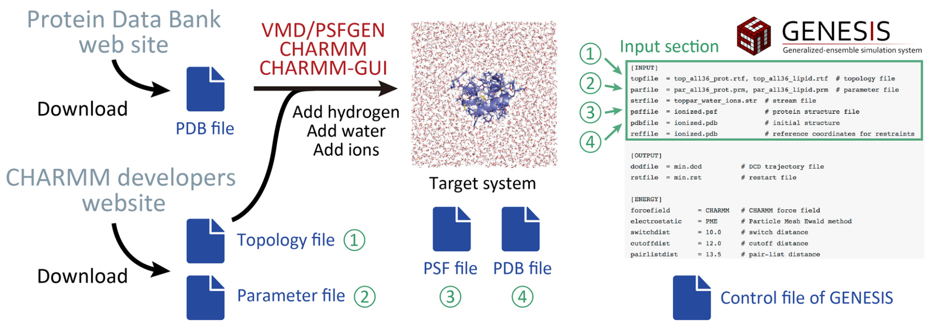
Force field	Input files	Setup tool
CHARMM	top, par, psf, pdb (or crd), str	VMD, PSFGEN, CHARMM-GUI, CHARMM
AMBER	prmtop, pdb, (or ambcrd)	LEaP
KB Gō model	top, par, psf, pdb	MMTSB server
All-atom Gō model	grotop, grocrd (or pdb)	SMOG server, SMOG2

4.1.1 CHARMM force field

One of the commonly used parameters for biomolecules is the CHARMM force field, which was originally developed by the Karplus group at the Harvard University [7]. The users can obtain the files that contain the force field parameters from the CHARMM group's web site. At this momemt, the latest version of the CHARMM force field is C36m [13]. In the download file, there are topology and parameter files (e.g., top_all36_prot.rtf and par_all36m_prot.prm).

In order to run the MD simulation with the CHARMM force field, the users have to additionally make a new file that holds the information about the atom connectivity of the "whole" target system. Note that the topology file (e.g., top_all36_prot.rtf) does not contain such information, because it is designed to generally define the topology of proteins by dealing with the 20 amino acid residues as "fragments". In order to hold the topology information of the target system, the users will create a "PSF" file (protein structure file). It is commonly used in other MD software, and can be generated from the PDB and topology files by using VMD/PSFGEN [14], CHARMM-GUI [15], or the CHARMM program [2].

When the PSF file is created, "processed PDB" file is also obtained, where the atom name or residue name might be changed from those in the original PDB file. The users must use this PDB file as the input of the MD simulation, because it has a consistency with the information in PSF. Consequently, the users need four files (processed PDB, parameter, topology, and PSF) as the inputs of **GENESIS**. These files are specified in the [INPUT] section of the control file of **GENESIS**.



```
[INPUT]
topfile  = top_all36_prot.rtf, top_all36_lipid.rtf  # topology file
parfile  = par_all36_prot.prm, par_all36_lipid.prm  # parameter file
strfile  = toppar_water_ions.str  # stream file
psffile  = ionized.psf            # protein structure file
pdbfile  = ionized.pdb            # initial structure
reffile  = ionized.pdb            # reference coordinates for restraints

[OUTPUT]
dcdfile  = min.dcd          # DCD trajectory file
rstfile  = min.rst          # restart file

[ENERGY]
forcefield       = CHARMM   # CHARMM force field
electrostatic    = PME      # Particle Mesh Ewald method
switchdist       = 10.0     # switch distance
cutoffdist       = 12.0     # cutoff distance
pairlistdist     = 13.5     # pair-list distance
```

4.1.2 AMBER force field

The AMBER force field has been also commonly used for the MD simulations of biomolecules, which was originally developed by the Kollman group at the University of California, San Francisco [16]. **GENESIS** can deal with the AMBER force fields. Basic scheme to prepare the input files for **GENESIS** is similar to that in the case of CHARMM. The users utilize the LEaP program in AmberTools [1]. LEaP has a similar function to PSFGEN. After building the target system using LEaP, the users obtain PRMTOP, CRD, and PDB files. PRMTOP contains the information about parameter and topology of the target system, and CRD and PDB include the coordinates of atoms in the initial structure. **GENESIS** uses these files as the inputs.

4.1.3 Other force fields

GENESIS can deal with coarse-grained (CG) models such as the Gō model [17] and MARTINI [18]. In this case, the users again use external setup tools to build the system and prepare the parameter and topology files. For the all-atom Gō model [19], the users use the SMOG server [20] or SMOG2 program [21], which generates grotop and grocrd files. The grotop file contains the information about parameter and topology, and the grocrd file includes the coordinates of the initial structure, both of which are the file formats used in the GROMACS program. For the Karanicolas-Brooks (KB) Gō model [22] [23], the users use the MMTSB server [24], which generates par, top, pdb, and psf files.

4.2 General input files

topfile

CHARMM topology file containing information about atom connectivity of residues and other molecules. For details on the format, see the CHARMM web site [25].

parfile

CHARMM parameter file containing force field parameters, e.g. force constants and equilibrium geometries.

strfile

CHARMM stream file containing both topology information and parameters.

psffile

CHARMM/X-PLOR ‘psffile‘ containing information of the system such as atomic masses, charges, and atom connectivities.

prmtopfile

AMBER 'PARM' or 'prmtop' file (AMBER7 or later format) containing information of the system such as atomic masses, charges, and atom connectivities. For details about this format, see the AMBER web site [26].

grotopfile

Gromacs 'top' file containing information of the system such as atomic masses, charges, atom connectivities. For details about this format, see the Gromacs web site [27].

pdbfile

Coordinates file in the PDB format. If *rstfile* is also specified in the **[INPUT]** section, coordinates in *pdbfile* are replaced with those in *rstfile*.

crdfile

Coordinates file in the CHARMM format. If *pdbfile* is also specified in the **[INPUT]** section, coordinates in *crdfile* are NOT used. However, if *pdbfile* is not specified, coordinates in *crdfile* are used. If *rstfile* is further specified, coordinates in *rstfile* are used.

ambcrdfile

Coordinates file in the AMBER format (ascii). If *pdbfile* is also specified in the **[INPUT]** section, coordinates in *ambcrdfile* are NOT used. However, if *pdbfile* is not specified, coordinates in *ambcrdfile* are used. If *rstfile* is further specified, coordinates in *rstfile* are used.

grocrdfile

Coordinates file in the GROMACS format (.gro file). If *pdbfile* is also specified in the **[INPUT]** section, coordinates in *grocrdfile* are NOT used. However, if *pdbfile* is not specified, coordinates in *grocrdfile* are used. If *rstfile* is further specified, coordinates in *rstfile* are used. Note that velocites and simulation box size in *grocrdfile* are NOT used.

rstfile

Restart file in the GENESIS format. This file contains atomic coordinates, velocities, simulation box size, and other variables which are essential to restart the simulation continuously. If *rstfile* is specified in the **[INPUT]** section, coordinates in *pdbfile*, *crdfile*, *grocrdfile*, or *ambcrdfile* are replaced with those in *rstfile*. The box size specified in the **[BOUNDARY]** seciton is also overwritten. Note that *pdbfile*, *crdfile*, *grocrdfile*, or *ambcrdfile* should be still specified in the **[INPUT]** section, even if *rstfile* is specified.

Note that the file format of *rstfile* was changed after ver. 1.1.0. The *rst_upgrade* tool enables us to change the old format used in ver. 1.0.0 or older to the new one.

4.3 Input files for restraint

reffile

Reference coordinates (PDB file format) for positional restraints and coordinate fitting. This file should contain the same total number of atoms as *pdbfile*, *crdfile*, *ambcrdfile*, or *grocrdfile*.

ambreffile

Reference coordinates ('amber crd' file format) for positional restraints and coordinate fitting. This file should contain the same total number of atoms as *pdbfile* or *ambcrdfile*.

groreffile

Reference coordinates ('gro' file format) for positional restraints and coordinate fitting. This file should contain the same total number of atoms as *pdbfile* or *grocrdfile*.

modefile

Principal modes used with principal component (PC) restraints. This file contains only single column ascii data. The XYZ values of each atom's mode vector are stored from the low-index modes.

localresfile (for **SPDYN** only)

This file defines restraints to be applied in the system. If you are not an expert of GENESIS, we strongly recommend you to simply use the **[RESTRAINTS]** section for restraint instead of using **localresfile**.

In **localresfile**, only bond, angle, and dihedral angle restraints can be defined. In addition, selected atoms in **localresfile** must exist in the same cell in the domain decomposition scheme. The restraint energy calculated for the lists in **localresfile** is NOT explicitly displayed in the log file. Instead, the local restraint energy is hidden in the conventional bond, angle, and dihedral angle energy terms of the log file.

The restraint potentials defined in **localresfile** are given by harmonic potentials:

$U(r) = k\ (r - r_0)^2$ for bonds

$U(\theta) = k\ (\theta - \theta_0)^2$ for bond angles

$U(\phi) = k\ (\phi - \phi_0)^2$ for dihedral angles

Here, r, θ, and ϕ are bond distance, angle, and dihedral angles, respectively; subscript 0 denotes their reference values; and k is the force constant.

The syntax in **localresfile** is as follows:

```
[BOND/ANGLE/DIHEDRAL]     atom atom [atom [atom]]  k r0
```

The users must carefully specify the atom index in this file. The atom indexes in **localresfile** must be consistent with those in the other input files such as **psffile**.

The following is an example of **localresfile**:

```
BOND      139 143          2.0 10.0
ANGLE     233 231 247      3.0 10.0
DIHEDRAL   22  24  41  43  2.0 10.0
```

4.4 Input files for REMD and RPATH simulations

In the REMD or RPATH simulations, input files (mainly coordinates and restart files) should be prepared for each replica. In GENESIS, we can easily specify those multiple files in the **[INPUT]** section. If we include '{}' in the input filename, {} is automatically replaced with the replica index. For example, in the case of REMD simulations with 4 replicas, we prepare input_1.pdb, input_2.pdb, input_3.pdb, and input_3.pdb, and specify `pdbfile = input_{}.pdb` in the **[INPUT]** section. This rule is also applicable to the restart filename.

fitfile (for RPATH only; GENESIS 1.1.5 or later)

Reference coordinates for structure fitting. This file is only used in the string method. For other cases (MD, MIN, or REMD), *reffile*, *groreffile*, or *ambreffile* is used for reference coordinates for fitting, and this *fitfile* is simply ignored, even if it is specified in the **[INPUT]** section.

4.5 Examples

MD simulations of proteins in explicit solvent with the CHARMM36m force field:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
```

MD simulations with positional restraint:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
reffile = ../build/input.pdb
```

MD simulations of membrane proteins with the CHARMM36m force field:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf, ../toppar/top_all36_lipid.rtf
parfile = ../toppar/par_all36m_prot.prm, ../toppar/par_all36_lipid.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
```

In this case, we specify multiple top and par files for proteins and lipids separated by commas.

If one line becomes very long, backslash ” \ ” can be used as a line continuation character:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf, \
          ../toppar/par_all36_na.prm,   \
          ../toppar/top_all36_lipid.rtf
parfile = ../toppar/par_all36m_prot.prm, \
          ../toppar/top_all36_na.rtf,    \
          ../toppar/par_all36_lipid.prm
strfile = ../toppar/toppar_water_ions.str
```

(continues on next page)

(continued from previous page)

```
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
```

MD simulations with the AMBER force field:

```
[INPUT]
prmtopfile = ../build/input.prmtop
ambcrdfile = ../build/input.crd
```

MD simulations with the all-atom Gō model:

```
[INPUT]
grotopfile = ../build/input.top
grocrdfile = ../build/input.gro
```

In this case, we specify grotop and grocrd files obtained from the SMOG server or SMOG2 software.

REMD simulations starting from the same initial structure:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
```

REMD simulations starting from different initial structures:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input_rep{}.pdb
```

REMD simulations with restarting:

```
[INPUT]
topfile = ../toppar/top_all36_prot.rtf
parfile = ../toppar/par_all36m_prot.prm
strfile = ../toppar/toppar_water_ions.str
psffile = ../build/input.psf
pdbfile = ../build/input.pdb
rstfile = run_rep{}.rst
```

CHAPTER

FIVE

OUTPUT SECTION

GENESIS yields trajectory data (coordinates and velocities) in the *DCD* file format regardless of the force field or MD algorithm. **GENESIS** can also generate a restart file (*rstfile*) during or at the end of the simulation, which can be used to restart and extend the simulation continuously. Output frequency of each file (e.g., crdout_period and velout_period) is specified in the **[DYNAMICS]** section in the case of the MD, REMD, and RPATH simulations, or **[MINIMIZE]** section in the case of the energy minimization.

5.1 General output files

dcdfile

Filename for the coordinates trajectory data. Coordinates are written in the DCD format, which is commonly used in various MD software such as CHARMM and NAMD. The filename must be given in the case of `crdout_period > 0`. However, if `crdout_period = 0` is specified in the control file, no **dcdfile** is generated, even if the filename is specified in the **[OUTPUT]** section.

dcdvelfile

Filename for the velocity trajectory data. Velocities are written in the DCD format. The filename must be given in the case of `velout_period > 0`. However, if `velout_period = 0` is specified in the control file, no **dcdvelfile** is generated, even if the filename is specified in the **[OUTPUT]** section.

rstfile

Filename for the restart data. The rstfile contains coordinates, velocities, simulation box size, and so on. This file can be used to extend the simulation continuously. In addition, it can be used to switch the simulation algorithms (e.g., from minimization to MD, from MD to REMD, from REMD to minimization, etc) The filename must be given in the case of `rstout_period > 0`. However, if `rstout_period = 0` is specified in the control file, no **rstfile** is generated, even if the filename is specified in the **[OUTPUT]** section.

pdbfile (for ATDYN only)

Filename for the restart PDB file. This file is updated every `rstout_period` steps.

5.2 Output files in REMD and RPATH simulations

When the user performs REMD or RPATH simulations, the user must include '{}' in the output filename. This {} is automatically replaced with the replica index.

remfile (only for REMD simulations)

This file contains parameter index data from the REMD simulation, which is written for each replica every `exchange_period` steps. This is used as an input file for the *remd_convert* tool to sort the coordinates trajectory data by parameters. The filename must contain '{}', which is automatically replaced with the replica index. Note that the information about the parameter index as well as replica index in the entire REMD simulation is written in the standard (single) output file (see online Tutorials).

logfile (only for REMD and RPATH simulations)

This file contains the energy trajectory data from the REMD or RPATH simulations, which is written for each replica every `exchange_period` steps. This is used as an input file for the *remd_convert* tool to sort the coordinates trajectory data by parameters. The filename must contain '{}', which is automatically replaced with the replica index.

rpathfile (only for RPATH simulations)

This file contains the trajectory of image coordinates in the string method, which are reference values used in the restraint functions. Columns correspond to the collective variables, and rows are time steps. This data is written with the same timing as the *dcdfile*. For details, see *RPATH section*.

5.3 Output file in GaMD simulations

gamdfile

This file provides GaMD parameters determined during the GaMD simulation. The filename must be given in the case of `update_period > 0` in **[GAMD]** section. The GaMD simulation updates its parameters every `update_period`, and then updates parameters are output to **gamdfile**. The file includes the maximum, minimum, average, and deviation of the total potential or dihedral potential, which are calculated within the interval `update_period`.

5.4 Output file in Vibrational analysis

minfofile

This file provides the coordinates and normal mode vectors of the molecules specified for vibrational analysis. It is used in **SINDO** for visualizing the vibrational motion. It is also an input file to start anharmonic vibrational calculations. See *Vibration section* for the vibrational analysis.

5.5 Output file in FEP simulations

fepfile

This file provides energy differences between adjacent windows in FEP simulations. The filename must be given in the case of `fepout_period > 0` in **[ALCHEMY]** section. However, if `fepout_period = 0` is specified in the control file, no **fepfile** is generated, even if the filename is specified in the **[OUTPUT]** section. If FEP/λ-REMD simulations are performed (i.e., both **[REMD]** and **[ALCHEMY]** sections are specified in the control file), the filename must contain '{}', which is automatically replaced with the replica index.

5.6 Examples

For normal MD simulations:

```
[OUTPUT]
dcdfile = run.dcd
rstfile = run.rst
```

For REMD simulations:

```
[OUTPUT]
logfile = run_rep{}.log
dcdfile = run_rep{}.dcd
remfile = run_rep{}.rem
rstfile = run_rep{}.rst
```

CHAPTER

SIX

ENERGY SECTION

6.1 Force fields

In general, potential energy function is given by:

$$
\begin{aligned}
E(r) = & \sum_{\text{bond}} K_b (b - b_0)^2 + \sum_{\text{angle}} K_\theta (\theta - \theta_0)^2 \\
& + \sum_{\text{dihedral}} K_\phi (1 + \cos(n\phi - \delta)) + \sum_{\text{improper}} K_{\phi_i} (\phi_i - \phi_{i,0})^2 \\
& + \sum_{\text{nonbond}} \epsilon \left[\left(\frac{R_{min,ij}}{r_{ij}} \right)^{12} - 2 \left(\frac{R_{min,ij}}{r_{ij}} \right)^{6} \right] + \sum_{\text{nonbond}} \frac{q_i q_j}{\epsilon_1 r_{ij}}
\end{aligned}
$$

where K_b, K_θ, K_ϕ, and K_{ϕ_i} are the force constant of the bond, angle, dihedral angle, and improper dihedral angle term, respectively, and b_0, θ_0, ϕ_0, and $\phi_{i,0}$ are corresponding equilibrium values. δ is a phase shift of the dihedral angle potential, ϵ is a Lennard-Jones potential well depth, $R_{min,ij}$ is a distance of the Lennard-Jones potential minimum, q_i is an atomic charge, ϵ_1 is an effective dielectric constant, and r_{ij} is a distance between two atoms. The detailed formula and parameters in the potential energy function depend on the force field and molecular model.

---

forcefield *CHARMM / CHARMM19 / AMBER / GROAMBER / GROMARTINI / KBGO / CAGO / AAGO*

Default : CHARMM

Type of the force field used for energy and force calculation. For the AMBER force field, the scheme used in the GROMACS program package is availble in addition to that used in the AMBER package. In this case, calculation for the dispersion correction term and truncation of the non-bonded energy term are different between AMBER and GROMACS.

- **CHARMM**: CHARMM force field with the all-atom model (CHARMM22, 27, 36, 36m) [7] [28] [29] [30]
- **CHARMM19**: CHARMM force field with the united-atom model (**ATDYN** only)
- **AMBER**: AMBER force field with the original AMBER scheme [6]
- **GROAMBER**: AMBER force field with the GROMACS scheme
- **GROMARTINI**: MARTINI model [18] [31]
- **KBGO**: model by Karanicolas and Brooks [22] [23] (**ATDYN** only)
- **CAGO**: Cα Gō model [32] (**ATDYN** only)
- **AAGO**: All-atom Gō model [19]

6.2 Non-bonded interactions

Calculation of the non-bonded interaction is the most time consuming part in MD simulations. Computational time for the non-bonded interaction terms without any approximation is proportional to $O(N^2)$. To reduce the computational cost, a cut-off approximation is introduced, where the energy and force calculation is truncated at a given cut-off value (keyword *cutoffdist*). Simple truncation at the cut-off distance leads to discontinuous energy and forces. So it is necessary to introduce a polynomial function (so called *switching function*) that smoothly turn off the interaction from another given value (so called *switch cut-off*), which is generally applied to the van der Waals interactions (keyword *switchdist*). There are two kinds of switching: "potential switch" and "force switch". In **GENESIS**, potential switching is turned on as the default. However, in the case of the AMBER force field, potential switching is still turned off, since the original AMBER program package is not using the potential switching. To turn on the "force switching", `vdw_force_switch=YES` must be specified. Note that the cut-off scheme for the electrostatic energy term is different from that for the van der Waals energy term, where the former uses a shift function. Such shift is turned on when `Electrostatic=Cutoff` is specified.

---

electrostatic *CUTOFF / PME*

Default : PME

- **CUTOFF**: Non-bonded interactions including the van der Waals interaction are just truncated at *cutoffdist*.
- **PME**: Particle mesh Ewald (PME) method is employed for long-range interactions. This option is only availabe in the periodic boundary condition.

switchdist *Real*

Default : 10.0 (unit : Å)

Switch-on distance for nonbonded interaction energy/force quenching. If *switchdist* is set to be equal to *cutoffdist*, switching can be turned off. Switching scheme depends on the selected force field, *vdw_shift*, and *vdw_force_switch* parameters. In the case of AMBER force field, this switching must be disabled, because the switching function is not available. In the case of "forcefield = GROMARTINI" and "electrostatic = CUTOFF", *switchdist* is used only in the van der Waals potential energy. The switching-on distance for the electrostatic energy is automatically defined as 0.0.

cutoffdist *Real*

Default : 12.0 (unit : Å)

Cut-off distance for the non-bonded interactions. This distance must be larger than *switchdist*, while smaller than *pairlistdist*. In the case of the AMBER force field, this value must be equal to *switchdist*.

pairlistdist *Real*

Default : 13.5 (unit : Å)

Distance used to make a Verlet pair list for non-bonded interactions [33]. This distance must be larger than *cutoffdist*.

dielec_const *Real*

Default : 1.0

Dielectric constant of the system. Note that the distance dependent dielectric constant is not availabe in **GENESIS**.

vdw_force_switch *YES / NO*

Default : NO

This paramter determines whether the force switch function for van der Waals interactions is employed or not. [34] The users must take care about this parameter, when the CHARMM force field is used. Typically, "vdw_force_switch=YES" should be specified in the case of CHARMM36.

vdw_shift *YES / NO*

Default : NO

This parameter determines whether the energy shift for the van der Waals interactions is employed or not. If it is turned on, potential energy at the cut-off distance is shifted by a constant value so as to nullify the energy at that distance, instead of the default smooth quenching function. This parameter is available only when "forcefield = GROAMBER" or "forcefield = GROMARTINI".

dispersion_corr *NONE / ENERGY / EPRESS*

Default : NONE (automatically set to **EPRESS** in the case of AMBER)

This parameter determines how to deal with the long-range correction about the cut-off for the van der Waals interactions. Note that the formula used for the correction is different between the GROMACS and AMBER schemes. In the case of the CHARMM force filed, "dispersion_corr=NONE" is always used.

- **NONE**: No correction is carried out.
- **ENERGY**: Only energy correction is carried out.
- **EPRESS**: Both energy and internal pressure corrections are carried out.

contact_check *YES / NO*

Default : NO

If this parameter is set to *YES*, length of all covalent bonds as well as distance between non-bonded atom pairs are checked at the begining of the simulation. If long covalent bonds or clashing atoms are detected, those atom indexes are displayed in the log file. If *contact_check* is turned on, *nonb_limiter* is also automatically enabled. If the users want to turn on only "contact_check", please specify "contact_check = YES" and "nonb_limiter = NO" explicitly. Note that this contact_check does not work in the parallel-io scheme. If you are using **SPDYN**, please see also *structure_check*.

structure_check *NONE / FIRST / DOMAIN* (**SPDYN** only)

Default : NONE

If this parameter is set to FIRST or DOMAIN, length of all covalent bonds as well as distance between non-bonded atom pairs are checked at the begininig or during the simulation. This option is similar to *contact_check*, but has an improved capability when the parallel-io scheme is employed. In **SPDYN**, we recommend the users to use this option instead of *contact_check*. Since the structure check spends additional computational time, the users had better turn off this option in the production run.

- **NONE**: Do not check the structure

- **FIRST**: Check the structure only at the beginning of the simulation
- **DOMAIN**: Check the structure whenever the pairlist is updated

nonb_limiter *YES / NO*

Default : NO (automatically set to be equal to **contact_check**)

If this parameter is set to *YES*, large force caused by the atomic clash is suppressed during the simulation. Here, the atomic clash can be defined by *minimum_contact* (see below). If "contact_check = YES" is specified, this parameter is automatically set to "YES". If the users want to turn on only "contact_check", please specify "contact_check = YES" and "nonb_limiter = NO" explicitly. This option is basically useful for the energy minimization or equilibration of the system. However, we strongly recommend the users to turn off this option in the production run, because suppression of large forces is an "unphysical" manipulation to avoid unstable simulations.

minimum_contact *Real*

Default : 0.5 (unit : Å)

This parameter defines the clash distance, when `contact_check = YES` is specified. If the distance between the non-bonded atoms is less than this value, energy and force are computed using this distance instead of the actual distance.

6.3 Particle mesh Ewald method

Electrostatic energy in the conventional Ewald sum method is expressed as:

$$E_{elec} = \sum_{i<j} \frac{q_i q_j}{\epsilon_1} \frac{\mathrm{erfc}(\alpha r_{ij})}{r_{ij}} + \frac{2\pi}{V} \sum_{|\mathbf{G}|^2 \neq 0} \frac{\exp(-|\mathbf{G}|^2/4\alpha^2)}{|\mathbf{G}|^2} \sum_{ij} \frac{q_i q_j}{\epsilon_1} \exp(i\mathbf{G}\cdot\mathbf{r}_{ij}) - \sum_{ij} \frac{q_i q_j}{\epsilon_1} \frac{\alpha}{\sqrt{\pi}}$$

Here, the cut-off scheme can be used for the first term, because it decreases rapidly as distance between atoms increases. The third term is so called *self-energy*, and is calculated only once. The second term can be rewritten as:

$$\sum_{|\mathbf{G}|^2 \neq 0} \frac{\exp(-|\mathbf{G}|^2/4\alpha^2)}{|\mathbf{G}|^2} |\mathbf{S}(\mathbf{G})|^2$$

where the structure factor $\mathbf{S}(\mathbf{G})$ is defined as:

$$\mathbf{S}(\mathbf{G}) = \sum_i q_i \exp(i\mathbf{G}\cdot\mathbf{r}_i)$$

We cannot employ fast Fourier transformation (FFT) for the calculation of $\mathbf{S}(\mathbf{G})$ since atomic positions are usually not equally spaced. In the smooth particle mesh Ewald (PME) method [35] [36], this structure factor is approximated by using cardinal B-spline interpolation as:

$$\mathbf{S}(\mathbf{G}) = \sum_i q_i \exp(i\mathbf{G}\cdot\mathbf{r}_i) \approx b_1(G_1)b_2(G_2)b_3(G_3)\mathbf{F}(\mathbf{Q})(G_1, G_2, G_3)$$

where $b_1(G_1)$, $b_2(G_2)$, and $b_3(G_3)$ are the coefficients brought by the cardinal B-spline interpolation of order n and $\mathbf{Q}$ is a 3D tensor obtained by interpolating atomic charges on the grids. Since this $\mathbf{Q}$ has equally spaced structure, its Fourier transformation, $\mathbf{F}(\mathbf{Q})$, can be calculated by using FFT in the PME method.

---

pme_alpha *Real or auto*

Default : auto

Exponent of complementary error function. If `pme_alpha=auto` is specified, the value is automatically determined from *cutoffdist* and *pme_alpha_tol*.

Note: The default of pme_alpha was 0.34 in GENESIS ver. 1.1.0 or former.

pme_alpha_tol *Real*

Default : 1.0e-5

Tolerance to be used for determining *pme_alpha*, when `pme_alpha=auto` is specified.

pme_nspline *Integer*

Default : 4

B-spline interpolation order used for the evaluation of $b_1(G_1)$, $b_2(G_2)$, $b_3(G_3)$, and $\mathbf{Q}$. The order must be $>= 3$.

pme_max_spacing *Real*

Default : 1.2 (unit : Å)

Max PME grid size used in the automatic grid number determination This parameter is used only when *pme_ngrid_x*, *pme_ngrid_y*, and *pme_grid_z* are not given in the control file.

pme_ngrid_x *Integer*

Default : N/A (Optional)

Number of FFT grid points along x dimension. If not specified, program will determine an appropriate number of grids using `pme_max_spacing`.

pme_ngrid_y *Integer*

Default : N/A (Optional)

Number of FFT grid points along y dimension. If not specified, program will determine an appropriate number of grids using `pme_max_spacing`.

pme_ngrid_z *Integer*

Default : N/A (Optional)

Number of FFT grid points along z dimension. If not specified, program will determine an appropriate number of grids using `pme_max_spacing`.

pme_multiple *YES/NO* (**ATDYN** only)

Default : NO

IF pme_multiple is set to YES, MPI processes are divided into two groups to compute the PME real and reciprocal parts individually.

pme_mul_ratio *Integer* (**ATDYN** only)

Default : 1

Ratio of the MPI processors for real and reciprocal PME term computations (only used when "PME_multiple=YES" is specified).

FFT_scheme *1DALLGATHER / 1DALLTOALL / 2DALLTOALL* (**SPDYN** only)

Default : 1DALLTOALL

This is a highly advanced option concerning reciprocal space calculations. Users usually don't need to change this option. See ref [37] for details.

Note: Both of **ATDYN** and **SPDYN** use OpenMP/MPI hybrid parallel fast Fourier transformation library, FFTE [38]. The number of PME grid points must be multiples of 2, 3, and 5 due to the restriction of this library. Moreover, in **SPDYN**, there are several additional rules, which depends on the number of processes, in PME grid numbers. In **SPDYN**, we first define domain numbers in each dimension such that product of them equals to the total number of MPI processors. Let us assume that the domain numbers in each dimension are domain_x, domain_y, and domain_z. The restriction condition of the grid numbers are as follows:

(1) `pme_ngrid_x` should be multiple of (2* `domain_x`)

(2) `pme_ngrid_y` should be multiple of (2* `domain_y`)

(3) `pme_ngrid_z` should be multiple of `domain_z`

If the given number of PME grid points does not meet the above conditions, the program will automatically reassign suitable grid numbers which are larger than those written in the control input. In such cases, warning message will be shown in the log file.

6.4 Lookup table

The following keywords are relevant if CHARMM or AMBER force field is used. For a linearly-interpolating lookup table, table points are assigned at the unit interval of cut-off$^2/r^2$ and energy/gradients are evaluated as a function of $b_2(G_2)$ [11].

$$F(r^2) \approx F_{\text{tab}}(L) + t(F_{\text{tab}}(L+1) - F_{\text{tab}}(L))$$

where

$$L = \text{INT}(\text{Density} \times r_v^2/r^2)$$

and

$$t = \text{Density} \times r_v^2/r^2 - L$$

Linear interpolation is used if "Electrostatic=PME".

Density is the number of points per a unit interval. Lookup table using cubic interpolation is different from that of linear interpolation. In the case of cubic interpolation, monotonic cubic Hermite polynomial interpolation is used to impose the monotonicity of the energy value. Energy/gradients are evaluated as a function of r^2 [39] using four basis functions for the cubic Hermite spline : $h_{00}(t)$, $h_{10}(t)$, $h_{01}(t)$, $h_{11}(t)$

$$F(r^2) \approx F_{\text{tab}}(L-1)h_{00}(t) + \frac{F_{\text{tab}}(L-2)+F_{\text{tab}}(L-1)}{2}h_{10} + F_{\text{tab}}(L)h_{10}(t) + \frac{F_{\text{tab}}(L-1)+F_{\text{tab}}(L)}{2}h_{11}(t)$$

where

$$L = \text{INT}(\text{Density} \times r^2)$$

and

$$t = \text{Density} \times r^2 - L$$

Cubic iterpolation is used if “Electrostatic=Cutoff”.

6.5 Generalized Born/Solvent-Accessible Surface-Area model

Implicit solvent model is useful to reduce computational cost for the simulations of biomolecules [40]. The GB/SA (Generalized Born/Solvent accessible surface area) model is one of the popular implicit solvent models, where the electrostatic contribution to the solvation free energy (ΔG_{elec}) is computed with the GB theory [41], and the non-polar contribution (ΔG_{np}) is calculated from the solvent accessible surface area [42]. In the GB theory, solvent molecules surrounding the solute are approximated as a continuum that has the dielectric constant of ~80. To date, various GB models have been developed. In GENESIS, the OBC model [43] and LCPO method [44] are available in the calculations of the GB and SA energy terms, respectively. Note that the GB/SA model is implemented in **ATDYN** but NOT **SPDYN**. The solvation free energy is incorporated into the molecular mechanics potential energy function as an effective energy term, namely, $U = U_{\text{FF}} + \Delta G_{\text{elec}} + \Delta G_{\text{np}}$.

6.5.1 GB energy term

In the GB theory, the solvation free energy of solute is given by

$$\Delta G_{\text{elec}} = -\frac{1}{2}\left\{\frac{1}{\varepsilon_{\text{p}}} - \frac{\exp(-\kappa f_{ij})}{\varepsilon_{\text{w}}}\right\}\sum_{i,j}\frac{q_i q_j}{f_{ij}},$$

where ε_{p} and ε_{w} are the dielectric constants of solute and solvent, respectively, q_i and q_j are the partial charges on the *i*-th and *j*-th atoms, respectively. κ is the inverse of Debye length. f_{ij} is the effective distance between the *i*- and *j*-th atoms, which depends on the degree of burial of the atoms, and is given by

$$f_{ij} = \sqrt{r_{ij}^2 + R_i R_j \exp\left(\frac{-r_{ij}^2}{4R_i R_j}\right)}.$$

Here, r_{ij} is the actual distance between the *i*- and *j*-th atoms, and R_i is the effective Born radius of the *i*-th atom, which is typically estimated in the Coulomb field approximation by

$$\frac{1}{R_i} = \frac{1}{\rho_i} - \frac{1}{4\pi} \int_{\text{solute},r>\rho_i} \frac{1}{r^4} dV.$$

ρ_i is the radius of the *i*-th atom (mostly set to the atom's van der Waals radius), and the integral is carried out over the volume inside the solute but outside the *i*-th atom. In the case of an isolated ion, R_i is equal to its van der Waals radius. On the other hand, if the atom is buried inside a solute, R_i becomes larger, resulting in larger f_{ij}. In the OBC model, the effective Born radius is approximated as

$$\frac{1}{R_i} = \frac{1}{\tilde{\rho}_i} - \frac{1}{\rho_i} \tanh(\alpha\Psi_i - \beta\Psi_i^2 + \gamma\Psi_i^3),$$

where $\tilde{\rho}_i$ is defied as $\rho_i - \rho_0$ (intrinsic offset), and Ψ_i describes the dgree of burial of the solute atom, which is calculated from the pairwise descreening function: $\Psi_i = \tilde{\rho}_i \sum_j H(r_{ij})$ [45].

6.5.2 SA energy term

In general, the non-polar contribution to the solvation free energy is calculated by

$$\Delta G_{\text{np}} = \sum_i \gamma_i A_i,$$

where γ is the surface tension coefficient, and A_i is the surface area of the *i*-th atom. In the LCPO method, A_i is calculated from a linear combination of the overlaps between the neighboring atoms, given by

$$A_i = P_{1i} 4\pi R_i^2 + P_{2i} \sum_{j=1}^{n} A_{ij} + P_{3i} \sum_{j=1}^{n} \sum_{k=1}^{m} A_{jk} + P_{4i} \sum_{j=1}^{n} \left[A_{ij} \sum_{j=1}^{n} \sum_{k=1}^{m} A_{jk} \right].$$

P_{1-4} are the empirical parameters determined for each atom type, R_i is the radius of the *i*-th atom + probe radius (typically 1.4 Ang), and A_{ij} is the area of the *i*-th atom buried inside the *j*-th atom, given by

$$A_{ij} = 2\pi R_i \left(R_i - \frac{r_{ij}}{2} - \frac{R_i^2 - R_j^2}{2r_{ij}} \right)$$

where r_{ij} is the distance between the *i*- and *j*-th atoms.

---

implicit_solvent *NONE / GBSA* (**ATDYN** only)

Default : NONE

Turn on or off the GB/SA calculation.

- **NONE**: Do not perform GB/SA calculation.
- **GBSA**: Perform GB/SA calculation (Only available with the CHARMM all-atom force field in non-boundary condition ("type=NOBC" in the **[BOUNDARY]** section).

gbsa_eps_solvent *Real*

Default : 78.5

Dielectric constant of solvent ε_{w}.

gbsa_eps_solute *Real*

Default : 1.0

Dielectric constant of solute ε_{p}.

gbsa_alpha

Default : 1.0

The empirical parameter α in the equation for the effective Born radius calculation. "gbsa_alpha=0.8" for OBC1, and "gbsa_alpha=1.0" for OBC2.

gbsa_beta

Default : 0.8

The empirical parameter β in the equation for the effective Born radius calculation. "gbsa_beta=0.0" for OBC1, and "gbsa_beta=0.8" for OBC2.

gbsa_gamma

Default : 4.85

The empirical parameter γ in the equation for the effective Born radius calculation. "gbsa_gamma=2.91" for OBC1, and "gbsa_gamma=4.85" for OBC2.

gbsa_salt_cons

Default : 0.2 (unit : mol/L)

Concentration of the monovalent salt solution.

gbsa_vdw_offset

Default : 0.09 (unit : Å)

Intrinsic offset ρ_0 for the van der Waals radius.

gbsa_surf_tens

Default : 0.005 (unit : kcal/mol/Å^2)

Surface tension coefficient γ in the SA energy term.

Note: Debye length is calculated by $\kappa^{-1} = \sqrt{\varepsilon_0 \varepsilon_w k_B T / 2 N_A e^2 I}$, where *T* is automatically set to the target temperature specified in the **[DYNAMICS]** section. In the case of the energy minimization, *T* = 298.15 K is used. In the T-REMD simulations with the GB/SA model, each replica has an individual Debye length depending on the assigned temperature.

6.6 Examples

Simulation with the CHARMM36 force field in the periodic boundary condition:

```
[ENERGY]
forcefield       = CHARMM  # CHARMM force field
electrostatic    = PME     # use Particle mesh Ewald method
switchdist       = 10.0    # switch distance
cutoffdist       = 12.0    # cutoff distance
pairlistdist     = 13.5    # pair-list distance
vdw_force_switch = YES     # force switch option for van der Waals
pme_nspline      = 4       # order of B-spline in [PME]
pme_max_spacing  = 1.2     # max grid spacing allowed
```

Simulation with the AMBER force field in the periodic boundary condition:

```
[ENERGY]
forcefield       = AMBER   # AMBER force field
electrostatic    = PME     # use Particle mesh Ewald method
switchdist       = 8.0     # switch distance
cutoffdist       = 8.0     # cutoff distance
pairlistdist     = 9.5     # pair-list distance
pme_nspline      = 4       # order of B-spline in [PME]
pme_max_spacing  = 1.2     # max grid spacing allowed
```

Recommended options in the case of energy minimization (see *Minimize section*) for the initial structure with the CHARMM36 force field:

```
[ENERGY]
forcefield       = CHARMM  # CHARMM force field
electrostatic    = PME     # use Particle mesh Ewald method
switchdist       = 10.0    # switch distance
cutoffdist       = 12.0    # cutoff distance
pairlistdist     = 13.5    # pair-list distance
vdw_force_switch = YES     # force switch option for van der Waals
pme_nspline      = 4       # order of B-spline in [PME]
pme_max_spacing  = 1.2     # max grid spacing allowed
contact_check    = YES     # check atomic clash
nonb_limiter     = YES     # avoid failure due to atomic clash
minimum_contact  = 0.5     # definition of atomic clash distance
```

Simulations with the GB/SA implicit solvent model

```
[ENERGY]
forcefield       = CHARMM  # [CHARMM]
electrostatic    = CUTOFF  # [CUTOFF]
switchdist       = 23.0    # switch distance
cutoffdist       = 25.0    # cutoff distance
pairlistdist     = 27.0    # pair-list distance
implicit_solvent = GBSA    # Turn on GBSA calculation
gbsa_eps_solvent = 78.5    # solvent dielectric constant in GB
gbsa_eps_solute  = 1.0     # solute dielectric constant in GB
gbsa_salt_cons   = 0.2     # salt concentration (mol/L) in GB
gbsa_surf_tens   = 0.005   # surface tension (kcal/mol/A^2) in SA
vdw_force_switch = YES     # turn on van der Waals force switch
```

CHAPTER

SEVEN

DYNAMICS SECTION

7.1 Molecular dynamics simulations

In MD simulations, Newton's equation of motion ($F = ma$) is integrated numerically, where the force F is derived from the first derivative of the potential energy function with respect to the atomic position. To date, various integrators have been proposed. In the leap-frog algorithm, velocities are updated with

$$\mathbf{v}_i(t+\frac{\Delta t}{2}) = \mathbf{v}_i(t-\frac{\Delta t}{2}) + \frac{\Delta t}{m_i}\mathbf{F}_i(t),$$

and coordinates are updated with

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t \mathbf{v}_i(t+\frac{\Delta t}{2}).$$

In the velocity Verlet algorithm, coordinates and velocities are obtained at the same time. The velocities are updated with

$$\mathbf{v}_i(t) = \mathbf{v}_i(t-\Delta t) + \frac{\Delta t}{m_i}\frac{\mathbf{F}_i(t)+\mathbf{F}_i(t-\Delta t)}{2},$$

and the coordinates are updated with

$$\mathbf{r}_i(t+\Delta t) = \mathbf{r}_i(t) + \Delta t \mathbf{v}_i(t) + \frac{\Delta t^2}{m_i}\mathbf{F}_i(t).$$

In **ATDYN**, both leap-frog and velocity Verlet integrators are available. The multiple time step integrator (r-RESPA [46]) is also available in **SPDYN**. The users must pay attention to the **[ENSEMBLE]** section as well, because the algorithms that control the temperature and pressure are involved in the integrator. For details, see *Ensemble section*.

---

integrator *LEAP / VVER / VRES*

Default : LEAP

- **LEAP**: leap-frog integrator
- **VVER**: velocity Verlet integrator
- **VRES**: RESPA integrator (**SPDYN** only).

timestep *Real*

Default : 0.001 (unit : ps)

Time step in the MD run. In general, timestep can be extended to 2 fs or longer, when the SHAKE, RATTLE, or SETTLE algorithms are employed. (see *Constraints section*).

nsteps *Integer*

Default : 100

Total number of steps in one MD run. If “timestep=0.001” and “nsteps=1000000” are specified, the users can carry out 1-ns MD simulation.

eneout_period *Integer*

Default : 10

Output frequency for the energy data. The trajectories are written in the log file every **eneout_period** steps during the simulation. For example, if “timestep=0.001” and “eneout_period=1000” are specified, the energy is written every 1 ps.

crdout_period *Integer*

Default : 0

Output frequency for the coordinates data. The trajectories are written in the “dcdfile” specified in the **[OUTPUT]** section every **crdout_period** steps during the simulation.

velout_period *Integer*

Default : 0

Output frequency for the velocities data. The trajectories are written in the “dcdvelfile” specified in the **[OUTPUT]** section every **velout_period** steps during the simulation.

rstout_period *Integer*

Default : 0

Output frequency for the restart file. The restart information is written in the “rstfile” specified in the **[OUTPUT]** section every **rstout_period** steps during the simulation.

stoptr_period *Integer*

Default : 10

Frequency of removing translational and rotational motions of the whole system. Note that the rotational motion is not removed when the periodic boundary condition is employed. When you use positional restraints or RMSD restraints in the simulation, you may have to take care about removal of those motions. In some cases, such restraints can generate translational or rotational momentum in the system. If the momentum is frequently removed, the dynamics can be significantly disturbed.

nbupdate_period *Integer*

Default : 10

Update frequency of the non-bonded pairlist.

elec_long_period *Integer* (**VRES** in **SPDYN** only)

Default : 1

Frequency of long-range interaction calculation.

thermostat_period *Integer* (**VRES** in **SPDYN** only)

Default : 1

Frequency of thermostat integration. It must be multiple of **elec_long_period**.

barostat_period *Integer* (**VRES** in **SPDYN** only)

Default : 1

Frequency of barostat integration. It must be multiple of **thermostat_period**.

initial_time *Real*

Default : 0.0 (unit : ps)

Initial time of the MD run. Basically, you do not need to specify a certain value. This option is useful in the case of the restart MD run, because the initial time is reset to 0 ps.

iseed *Integer*

Default : automatically generated according to the current date and time

Seed for the pseudo-random number generator. This random number seed is used in the Langevin and Bussi thermostats (see *ensemble*). If **iseed** is not specified in the control file, it is automatically generated according to the current date and time. In the restart MD run, the random number seed is taken over from rstfile. However, if **iseed** value is specified in the control file in the restart run, it is alternatively used, and the seed in rstfile is neglected.

verbose *YES / NO*

Default : NO

Turn on or off the verbose output of the log information. For example, if “verbose=YES” is specified, virial and pressure of the system are written in the log file even in the case of the NVE or NVT ensemble.

7.2 Simulated annealing and heating

In simulated annelaing or heating protocol, the following keywords are additinally specified in the conventional MD simulation. In the protocol used in **GENESIS**, the target temperature is changed linearly. Note that the protocol is available only in the *LEAP* integrator.

---

annealing *YES / NO*

Default : NO

Turn on or off the simulated annealing or heating protocol.

anneal_period *Integer*

Default : 0

The target temperature is changed every **anneal_period** steps during the simulation.

dtemperature *Real*

Default : 0.0 (unit : Kelvin)

Magnitute of changes of the target temperature. If **dtemperature** > 0, the temperature is increased by **dtemperature** every **anneal_period** steps. If **dtemperature** < 0, the temperature is decreased.

7.3 Targeted MD and Steered MD simulations

In GENESIS, targeted MD (TMD) and steered MD (SMD) methods are available. These methods are useful to guide a protein structure towards a target. In SMD, restraint forces (or steering forces) are applied on the selected atoms, where the RMSD with respect to the target is changed during the MD simulation. The restraint force is calculated from the derivative of the RMSD restraint potential:

$$U = \frac{1}{2}k\left(RMSD(t) - RMSD_0(t)\right)^2$$

where $RMSD(t)$ is the instantaneous RMSD of the current coordinates from the target coordinates, and $RMSD_0$ is the target RMSD value. The target RMSD value is changed linearly from the initial to target RMSD values:

$$RMSD_0(t) = RMSD_{\text{initial}} + \frac{t}{T}\left(RMSD_{\text{final}} - RMSD_{\text{initial}}\right)$$

where T is the total MD simulation time. Targeted MD (TMD), originally suggested by J. Schlitter et al. [47], is different from SMD in that force constants are changed during MD simulations. If the users perform SMD, there is a possibility observing the large difference between the instantaneous RMSD and target RMSD. In TMD, force constants are given by Lagrangian multipliers to overcome the energy barrier between the instantaneous and target RMSDs. Therefore, the users could find trajectories where RMSD is almost identical to the target RMSD at each time. In **[SELECTION]** section, the users select atoms involved in RMSD calculations for SMD or TMD. Users should specify either *RMSD* or *RMSDMASS* (mass-weighted RMSD) in **[RESTRAINTS]** section to run TMD or SMD. In SMD, force constants defined in **[RESTRAINTS]** section are used, but force constants are automatically determined using Lagrangian multipliers during simulation in TMD.

---

target_md *YES / NO*

Default : NO

Turn on or off the targeted MD simulation.

steered_md *YES / NO*

Default : NO

Turn on or off the steered MD simulation.

initial_rmsd *Real*

Default : 0.0 (unit : Å)

Initial value of the reference rmsd. If not specified explicitly, it is calculated from the initial and reference structures.

final_rmsd *Real*

Default : 0.0 (unit : Å)

Final value of the reference rmsd.

Note: In the RMSD restraint, structure fitting scheme is specified in the **[FITTING]** section (see *Fitting section*). Since the default behavior was significantly changed in ver. 1.1.5 (no fitting applied on the default setting), the users of 1.1.4 or before must pay special attention on the fitting scheme. In versions of 1.1.4 or before, structure fitting is automatically applied for the atoms concerning restraint potential.

7.4 Examples

100-ps MD simulation with the velocity Verlet integrator with the timestep of 2 fs:

```
[DYNAMICS]
integrator        =   VVER  # velocity Verlet
nsteps            =  50000  # number of MD steps (100ps)
timestep          =  0.002  # timestep (2fs)
eneout_period     =    500  # energy output period (1ps)
crdout_period     =    500  # coordinates output period (1ps)
rstout_period     =  50000  # restart output period
nbupdate_period   =     10  # nonbond pair list update period
```

100-ps MD simulation with the RESPA integrator with the timestep of 2.5 fs:

```
[DYNAMICS]
integrator        =   VRES  # RESPA integrator
nsteps            =  40000  # number of MD steps (100ps)
timestep          = 0.0025  # timestep (2.5fs)
eneout_period     =    400  # energy output period (1ps)
crdout_period     =    400  # coordinates output period (1ps)
rstout_period     =  40000  # restart output period
nbupdate_period   =     10  # nonbond pair list update period
elec_long_period  =      2  # period of reciprocal space calculation
thermostat_period =     10  # period of thermostat update
barostat_period   =     10  # period of barostat update
```

The following is an example for simulated annelaing in the NVT ensemble (see *Ensemble section*), where the temperature is decreased from 500 K by 2 K every 250 steps in the 250,000-steps MD simulation (1 step = 2 fs). Thus, the temperature eventually reaches to 300 K during 50 ps. Note that heating or annealing is only available with the leap-frog integrator.

```
[DYNAMICS]
integrator       =   LEAP   # leap-frog integrator
nsteps           =  25000   # number of MD steps
timestep         =  0.002   # timestep (ps)
nbupdate_period  =     10   # nonbond pair list update period
annealing        =    YES   # simulated annealing
dtemperature     =   -2.0   # delta temperature
anneal_period    =    250   # temperature change period

```

(continues on next page)

(continued from previous page)

```
[ENSEMBLE]
ensemble          = NVT      # [NVT,NPT,NPAT,NPgT]
tpcontrol         = LANGEVIN # [BERENDSEN,LANGEVIN]
temperature       = 500.0    # initial temperature (K)
```

CHAPTER

EIGHT

MINIMIZE SECTION

8.1 Energy minimization

In the **[MINIMIZE]** section, the user can select methods for energy minimization. Currently, the steepest descent (SD) algorithm is available in **SPDYN** and **ATDYN**, and the limited memory version of Broyden-Fletcher-Goldfarb-Shano (LBFGS) is additionally available in **ATDYN**. Note that constraint algorithms such as SHAKE are not available in the energy minimization scheme in **GENESIS**. The energy minimization can be done with restraints (see *Restraints section*).

When the energy minimization is carried out for the initial structure, it is strongly recommended to use the option "contact_check=YES" in the **[ENERGY]** section (see *Energy section*). This is because the initial structure is usually artificial, and sometimes contains atomic clashes, where the distance between atoms is very short. Such strong interactions can generate huge forces on the atoms, resulting in unstable calculations, which might cause memory errors.

---

method *SD / LBFGS*

Default : LBFGS (for **ATDYN**), **SD** (for **SPDYN**)

Algorithm of minimization.

- **SD** : Steepest descent method
- **LBFGS** : Limited memory version of Broyden-Fletcher-Goldfarb-Shano method (**ATDYN** only)

nsteps *Integer*

Default : 100

Number of minimization steps.

eneout_period *Integer*

Default : 10

Frequency of energy outputs.

crdout_period *Integer*

Default : 0

Frequency of coordinates outputs.

rstout_period *Integer*

Default : 0

Frequency of restart file updates.

nbupdate_period *Integer*

Default : 10

Frequency of non-bonded pair-list updates

fixatm_select_index *Integer* (**ATDYN** only)

Default : N/A (all atoms are minimized)

Index of an atom group to be fixed during minimization. The index must be defined in **[SELECTION]** (see *Selection section*). For example, if the user specifies `fixatm_select_index = 1`, the reference atoms should be members of `group1` in the **[SELECTION]**.

tol_rmsg *Real* (**ATDYN** only)

Default : 0.36 (unit : kcal/mol/Å)

Tolerence of convergence for RMS gradient.

tol_maxg *Real* (**ATDYN** only)

Default : 0.54 (unit : kcal/mol/Å)

Tolerence of convergence for maximum gradient.

Note: In **ATDYN**, a minimization run stops when *both* RMSG and MAXG become smaller than the tolerence values.

8.2 Steepest descent method

force_scale_min *Real*

Default : 0.00005

Minimum value of the force scaling coefficient in the steepest descent method. This value is also used as the initial value of the scaling coefficient.

force_scale_max *Real*

Default : 0.0001

Maximum value of the force scaling coefficient in the steepest descent method.

8.3 LBFGS method

ncorrection *Integer*

Default : 10

Number of corrections to build the inverse Hessian.

lbfgs_bnd *YES / NO*

Default : YES

Set a boundary to move atoms in each step of minimization.

lbfgs_bnd_qmonly *YES / NO*

Default : NO

Set the boundary only to QM atoms.

lbfgs_bnd_maxmove *Real*

Default : 0.1 (unit : Å)

The maximum size of move in each step.

Note: LBFGS often makes a large move of atoms, especially, in the first few steps, and creates a distorted structure. Although this is rarely a problem in MM calculation, it may cause convergence problem in QM calculation. `lbfgs_bnd` prevents a huge move and crush of atoms by setting a maximum size of move. The size is set by `lbfgs_bnd_maxmove`.

8.4 Macro/micro-iteration scheme in QM/MM

In this scheme, the MM region is first minimized while holding the QM region fixed. This step is called micro-iteration. When the MM region reaches the minima (or the maximum number of steps), the whole system including the QM region is updated. This step is called macro-iteration. Then, the MM region is minimized again with the new QM region. The micro- and macro-iterations are repeated until the convergence is reached.

This scheme requires time-consuming QM calculations only in the macro-iteration. During the micro-iteration, ESP charges are used to represent the electrostatic interaction between QM and MM region. Therefore, it is by far more efficient than the usual scheme, and is recommended to use when ESP charges are available. Currently, this scheme works in combination with Gaussian.

The keywords in this subsection have no effect in MM calculations, of course.

---

macro *YES / NO*

Default : NO

Invoke macro/micro-iteration scheme if YES.

nsteps_micro *Integer*

Default : 100

Number of minimization steps for micro-iteration.

tol_rmsg_micro *Real*

Default : 0.27 (unit : kcal/mol/Å)

Tolerence of convergence for RMS gradient in micro-iteration.

tol_maxg_micro *Real*

Default : 0.41 (unit : kcal/mol/Å)

Tolerence of convergence for maximum gradient in micro-iteration.

macro_select_index *Integer*

Index of an atom group to be fixed in micro-iteration, and minimized in macro-iteration. The index must be defined in **[SELECTION]** (see *Selection section*). QM atoms are selected by default.

8.5 Examples

A 2,000-step energy minimization with the steepest descent method:

```
[MINIMIZE]
method           = SD      # Steepest descent
nsteps           = 2000    # number of minimization steps
eneout_period    =   50    # energy output period
crdout_period    =   50    # coordinates output period
rstout_period    = 2000    # restart output period
nbupdate_period  =   10    # nonbond pair list update period
```

An example of LBFGS optmization along with the macro/micro-iteration scheme:

```
[MINIMIZE]
method              = LBFGS
nsteps              = 500  # number of steps
eneout_period       = 5    # energy output period
crdout_period       = 5    # coordinates output period
rstout_period       = 5    # restart output period
nbupdate_period     = 1    # nonbond pair list update period
lbfgs_bnd           = yes  # set a boundary to move atoms
lbfgs_bnd_qmonly    = no   # set the boundary only to QM atoms
lbfgs_bnd_maxmove   = 0.1  # the max. size of move
macro               = yes  # switch macro/micro-iteration scheme
nsteps_micro        = 100  # number of steps of micro-iteration
```

CHAPTER

NINE

CONSTRAINTS SECTION

9.1 SHAKE/RATTLE algorithms

In the **[CONSTRAINTS]** section, keywords related to bond constraints are specified. In the leapfrog integrator, the SHAKE algorithm is applied for covalent bonds involving hydrogen [48]. In the velocity Verlet and multiple time-step integrators, not only SHAKE but also RATTLE are used [49]. Note that bond constraint between heavy atoms is not available currently.

---

rigid_bond *YES / NO*

Default : NO

Turn on or off the SHAKE/RATTLE algorithms for covalent bonds involving hydrogen.

shake_iteration *Integer*

Default : 500

Maximum number of iterations for SHAKE/RATTLE constraint. If SHAKE/RATTLE does not converge within the given number of iterations, the program terminates with an error message.

shake_tolerance *Real*

Default : 1.0e-10 (unit : Å)

Tolerance of SHAKE/RATTLE convergence.

hydrogen_type *NAME / MASS*

Default : NAME

This parameter defines how hydrogen atoms are detected. This parameter is ignored when *rigid_bond* = *NO*. Usually, the users do not need to take care about this parameter.

- **MASS** : detect hydrogen only based on the atomic mass. If the mass of an atom is less than *hydrogen_mass_upper_bound* and greater than 0, that atom is considered as a hydrogen.
- **NAME** : detect hydrogen based on the atom name, type, and mass. If the mass of an atom is less than *hydrogen_mass_upper_bound* and the name or type begins with 'h', 'H', 'd', or 'D', that atom is considered as a hydrogen.

atom name (type)	mass	*NAME*	*MASS*
HX	1.0	o	o
XX	1.0	x	o
HY	3.0	x	x
YY	3.0	x	x

o: treated as hydrogen, x: not treated as hydrogen. Here, we assumed *hydrogen_mass_upper_bound* 2.1.

hydrogen_mass_upper_bound *Real*

Default : 2.1

This parameter defines the upper limit of atomic mass to determine the hydrogen atom. For exmaple, if you define it as 3.0, the atom with the atomic mass less than 3.0 is treated as a hydrogen. You should write it in the case of hydrogen mass repartitioning scheme. This option is available in **GENESIS 1.2** or later.

9.2 SETTLE algorithm

fast_water *YES / NO*

Default : YES

Turn on or off the SETTLE algorithm for the constraints of the water molecules [50]. Although the default is "fast_water=YES", the users must specify "rigid_bond=YES" to use the SETTLE algorithm. If "rigid_bond=YES" and "fast_water=NO" are specified, the SHAKE/RATTLE algorithm is applied to water molecules, which is not computationally efficient.

water_model *expression or NONE*

Default : TIP3

Residue name of the water molecule to be rigidified in the SETTLE algorithm. In the case of the AMBER force field, "water_model = WAT" must be specified.

Note: TIP4P water model is availabe in **GENESIS 1.2** or later. In the case of using TIP4P water model, we regard it as rigid. In molecular dynamics simulations, please define **rigid_bond** and **fast_water** yes. In minimization, **[Constraints]** has not been used before, but now you can define **fast_water** yes when TIP4P water model is used, by regarding TIP4P water molecule rigid. However, please keep in mind that other parameters cannot be defined in minimizations, and constraints are not applied except water molecules. TIP4P water model can be used only in SPDYN.

9.3 LINCS algorithm

fast_bond *YES / NO* (**LEAP** integrator in **ATDYN** only)

Default : NO

Turn on or off the LINCS algorithm. To use the LINCS algorithm, "rigid_bond=YES" should be also specified.

lincs_iteration *Integer* (**ATDYN** only)

Default : 1

Number of iterations in the LINCS algorithm.

lincs_order *Integer* (**ATDYN** only)

Default : 4

Matrix expansion order in the LINCS algorithm.

9.4 Examples

In the case of the CHARMM force field:

```
[CONSTRAINTS]
rigid_bond   = YES    # Turn on SHAKE/RATTLE
fast_water   = YES    # Turn on SETTLE
```

In the case of the AMBER force field:

```
[CONSTRAINTS]
rigid_bond   = YES    # Turn on SHAKE/RATTLE
fast_water   = YES    # Turn on SETTLE
water_model = WAT     # residue name of the rigid water
```

Turn off all constraints in the system

```
[CONSTRAINTS]
rigid_bond   = NO
fast_water   = NO
```

CHAPTER

TEN

ENSEMBLE SECTION

10.1 Thermostat and barostat

In the **[ENSEMBLE]** section, the type of ensemble, temperature and pressure control algorithm, and parameters used in these algorithms (such as temperature and pressure) can be specified.

In the Langevin thermostat algorithm ("ensemble=NVT" with "tpcontrol=LANGEVIN"), every particles are coupled with a viscous background and a stochastic heat bath [51]:

$$\frac{d\mathbf{v}(t)}{dt} = \frac{\mathbf{F}(t) + \mathbf{R}(t)}{m} - \gamma\mathbf{v}(t)$$

where γ is the thermostat friction parameter (*gamma_t* keyword) and $\mathbf{R}(t)$ is the stochastic force. In the Langevin thermostat and barostat method ("ensemble=NPT" with "tpcontrol=LANGEVIN"), the equation of motion is given by [52]:

$$\begin{aligned}
\frac{d\mathbf{r}(t)}{dt} &= \mathbf{v}(t) + v_\epsilon \mathbf{r}(t) \\
\frac{d\mathbf{v}(t)}{dt} &= \frac{\mathbf{F}(t) + \mathbf{R}(t)}{m} - [\gamma_p + (1 + \frac{3}{f})v_\epsilon]\mathbf{v}(t) \\
\frac{dv_\epsilon(t)}{dt} &= [3V(P(t) - P_0(t)) + \frac{3K}{f} - \gamma_p v_\epsilon + R_p]/p_{mass}
\end{aligned}$$

where K is the kinetic energy, γ_p is the barostat friction parameter (*gamma_p* keyword), R_p is the stochastic pressure variable.

---

ensemble *NVE / NVT / NPT / NPAT / NPgT*

Default : NVE

Type of ensemble.

- **NVE**: Microcanonical ensemble.
- **NVT**: Canonical ensemble.
- **NPT**: Isothermal-isobaric ensemble.
- **NPAT**: Constant area A (XY), pressure along the normal (Z), temperature [53]. In this case, *isotropy* must be set to 'XY-FIXED' (see below).
- **NPgT**: Constant surface-tension γ (XY), pressure along the normal (Z), temperature [53]. In this case, *isotropy* must be set to 'SEMI-ISO' (see below).

temperature *Real*

> **Default : 298.15** (unit : Kelvin)
>
> Initial and target temperature.

pressure *Real*

> **Default : 1.0** (unit : atm)
>
> Target pressure in the NPT ensemble. In the case of the NPAT and NPgT ensembles, this is the pressure along the 'Z' axis.

gamma *Real*

> **Default : 0.0** (unit : dyn/cm)
>
> Target surface tension in NPgT ensemble.

tpcontrol *NO / BERENDSEN / LANGEVIN / BUSSI*

> **Default : NO**
>
> Type of thermostat and barostat. The availabe algorithm depends on the integrator.
>
> - **NO**: Do not use temperature/pressure control algorithm (for NVE only)
> - **BERENDSEN**: Berendsen thermostat/barostat [54]
> - **LANGEVIN**: Langevin thermostat/barostat [52]
> - **BUSSI**: Bussi's thermostat/barostat [55] [56]

integrator	ensemble	tpcontrol
LEAP	NVT	BERENDSEN, LANGEVIN
	NPT	BERENDSEN, LANGEVIN
	NPAT/NPgT	BERENDSEN, LANGEVIN
VVER	NVT	BERENDSEN, LANGEVIN, BUSSI
	NPT	LANGEVIN, BUSSI
	NPAT/NPgT	LANGEVIN
VRES	NVT	LANGEVIN, BUSSI
	NPT	LANGEVIN, BUSSI
	NPAT/NPgT	LANGEVIN

tau_t *Real*

> **Default : 5.0** (unit : ps)
>
> Temperature coupling time in the Berendsen and Bussi thermostats.

tau_p *Real*

> **Default : 5.0** (unit : ps)
>
> Pressure coupling time in the Berendsen and Bussi barostats.

compressibility *Real*

> **Default : 0.0000463** (unit : atm^{-1})
>
> Compressibility parameter in the Berendsen barostat.

gamma_t *Real*

Default : 1.0 (unit : ps^{-1})

Friction parameter of the Langevin thermostat.

gamma_p *Real*

Default : 0.1 (unit : ps^{-1})

Friction parameter of the Langevin barostat.

isotropy *ISO / ANISO / SEMI-ISO / XY-FIXED*

Default : ISO

Isotropy of the simulation system. This parameter specifies how X, Y, Z dimensions of the simulation box change in NPT, NPgT, and NPAT ensembles.

- **ISO**: X, Y, and Z dimensions are coupled together.
- **ANISO**: X, Y, and Z dimensions fluctuate independently.
- **SEMI-ISO**: X, Y, and Z dimensions fluctuate, where the ratio of X and Y dimensions are kept constant, and Z dimension can change independently [57]. This setting with NPT or NPAT or NPgT ensemble is expected to be useful for bio-membrane systems.
- **XY-FIXED**: X and Y dimensions are fixed, while Z dimension can change (NPAT only).

10.2 Examples

NVT ensemble with Bussi thermostat:

```
[ENSEMBLE]
ensemble    = NVT       # Canonical ensemble
tpcontrol   = BUSSI     # Bussi thermostat
temperature = 300.0     # target temperature (K)
```

NPT ensemble with isotropic pressure coupling:

```
[ENSEMBLE]
ensemble    = NPT       # Isothermal-isobaric ensemble
tpcontrol   = BUSSI     # Bussi thermostat and barostat
temperature = 300.0     # target temperature (K)
pressure    = 1.0       # target pressure (atm)
```

NPT ensemble with semi-isotropic pressure coupling, which is usually used for lipid bilayer systems:

```
[ENSEMBLE]
ensemble    = NPT       # Isothermal-isobaric ensemble
tpcontrol   = BUSSI     # Bussi thermostat and barostat
temperature = 300.0     # target temperature (K)
pressure    = 1.0       # target pressure (atm)
isotropy    = SEMI-ISO  # Ratio of X to Y is kept constant
```

NPAT ensemble:

```
[ENSEMBLE]
ensemble    = NPAT      # Constant area ensemble
tpcontrol   = BUSSI     # Bussi thermostat and barostat
temperature = 300.0     # target temperature (K)
pressure    = 1.0       # target normal pressure (atm)
isotropy    = XY-FIXED  # the system area is kept constant
```

NPγT ensemble:

```
[ENSEMBLE]
ensemble    = NPgT      # Constant surface-tension ensemble
tpcontrol   = BUSSI     # Bussi thermostat and barostat
temperature = 300.0     # target temperature (K)
pressure    = 1.0       # target normal pressure (atm)
gamma       = 200.0     # target surface tension (dyn/cm)
isotropy    = SEMI-ISO  # Ratio of X to Y is kept constant
```

CHAPTER

ELEVEN

BOUNDARY SECTION

11.1 Boundary condition

type *PBC / NOBC*

Default : PBC

Type of boundary condition.

- **PBC**: Periodic boundary condition (rectangular or cubic box)
- **NOBC**: Non-boundary condition (vacuum system). In **ATDYN**, **NOBC** is applicable to various force fields and models. However, in **SPDYN**, it can be used for only all-atom Gō model.

box_size_x *Real*

Default : N/A (unit : Å)

Box size along the x dimension.

box_size_y *Real*

Default : N/A (unit : Å)

Box size along the y dimension.

box_size_z *Real*

Default : N/A (unit : Å)

Box size along the z dimension.

Note: If the simulation system has a periodic boundary condition (**PBC**), the user must specify the box size in the control file (at the energy minimization stage in most cases). During the simulations, box size is saved in a restart file. If the restart file is used as an input of the subsequent simulation, the box size is overwritten with the restart information. Note that in this case the box size given in the control file is ignored.

11.2 Domain decomposition

domain_x *Integer*

Default : N/A (Optional) (**SPDYN** only)

Number of domains along the x dimension.

domain_y *Integer*

Default : N/A (Optional) (**SPDYN** only)

Number of domains along the y dimension.

domain_z *Integer*

Default : N/A (Optional) (**SPDYN** only)

Number of domains along the z dimension.

Note: If number of domains (domain_x, domain_y, and domain_z) are not specified in the control file, they are automatically determined based on the number of MPI processes. When the user specifies the number of domains explicitly, please make sure that the product of the domain numbers in each dimension (i.e., domain_x * domain_y * domain_z) is equal to the total number of MPI processes.

11.3 Spherical potential

In MD simulations with **NOBC**, molecules may evaporate from a system, and, once such an event happens, the molecule runs in the vacuum with constant velocity to infinity. Therefore, it is useful to set a potential which pulls the molecule back to the system.

In **ATDYN**, the users can set a spherical potential,

$$\begin{aligned} V &= k(r_i - r_b)^n \quad (r_i > r_b) \\ &= 0, \quad (r_i \le r_b) \end{aligned}$$

where k, n and r_b are a force constant, an exponent, and a radius of the sphere, respectively, and r_i is the distance between the i-th atom and the center of sphere,

$$r_i = |\mathbf{x}_i - \mathbf{x}_0|.$$

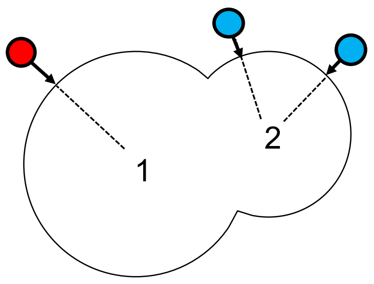


Fig. 11.1: An illustration of a combination of two spherical potentials (black thin circles), which pulls back atoms that are out of the range (gray) towards the center of sphere (1 and 2).

Multiple spheres with different centers and radii can be combined to construct the potential; for example, two spheres are combined in Fig. 11.1. The atoms that went out of the sphere (thin line) are pulled back to the nearest center; the red atom to center 1 and the blue atoms to center 2.

The coordinates of the center can be specified in two ways. The first is to set the center to a position of atoms in the initial structure (pdbfile) using **[SELECTOR]**. The other way is to directly specify coordinates of the center in the input. See the description of options and the examples below for details.

The following options are available to set the spherical potential:

---

spherical_pot *YES / NO*

> **Default : NO**
>
> If YES (with type=NOBC), use the spherical boundary potential.

constant *Real*

> **Default : 10.0** (unit : $\mathrm{kcal mol^{-1}}$)
>
> The force constant of the potential.

exponent *Integer*

> **Default : 2**
>
> The exponent of the potential.

nindex *Integer*

> **Default : 0**
>
> The number of index, used with center_select_index *N*.

center_select_index *N Integer*

> **Default : N/A**
>
> The index of center in **[SELECTION]**

nfunction *Integer*

> **Default : 0**
>
> The number of function, used with center *N*.

center *N Real* $\times 3$

> **Default : N/A**
>
> The xyz coordinates of the center.

radius *N*

> **Default : 0.0** (unit : Å)
>
> The radius of sphere.

fixatom *YES / NO*

> **Default : YES**
>
> Atoms out of the sphere in the input structure are fixed.

fixlayer *Real*

Default : 1.0 (unit : Å)

If fixatom = YES, atoms within this distance from the potential in the input structure are also fixed.

restart *YES / NO*

Default : YES

Use the information in the restart file.

Note: The information of the sphere and fixed atoms are saved in a restart file. If the information exists in rstfile, the options for the spherical potential in **[BOUNDARY]** will be ignored. If you want to re-set the potential, you need to specify **restart** = NO.

11.4 Examples

- Simulations in the gas-phase:

```
[BOUNDARY]
type        = NOBC       # non-periodic system
```

- Simulations with the periodic boundary condition, where the box size is set to 64 x 64 x 64. In this case, the user should not use a restart file as an input, because the box size in the control is overwritten with that in the restart file.

```
[BOUNDARY]
type        = PBC        # periodic boundary condition
box_size_x = 64.0       # Box size in the x dimension (Ang)
box_size_y = 64.0       # Box size in the y dimension (Ang)
box_size_z = 64.0       # Box size in the z dimension (Ang)
```

- Simulations with two spherical potentials around atom number 1 and 100 with a radius of 22.0 Angs.

```
[BOUNDARY]
type           = NOBC
spherical_pot  = yes
constant       = 2.0
exponent       = 2
nindex         = 1
center_select_index1 = 2
radius1        = 22.0
fix_layer      = 0.0
fixatom        = no

[SELECTION]
...
group2           = ano:1 or ano:100
```

Note: Be careful not to set too many spheres because it may slow down the performance. If you want to set the spheres around a protein, instead of specifying all atoms in a protein, select part of the atoms, for example, by

```
group2 = segid:PROA and an=CA
```

- Simulations with two spherical potentials. The center coordinates are explicitly set by **center1** and **center2**. With **fixatom** =yes and **fix_layer** =1.0 Angs, the atoms that are farther than 34 Angs from the centers are fixed.

```
[BOUNDARY]
type          = NOBC       # [PBC,NOBC]
spherical_pot = yes
constant      = 10.0
exponent      = 2
nfunctions    = 2
center1       =  17.0, 0.0, 0.0     # [x,y,z]
radius1       = 35.0
center2       = -17.0, 0.0, 0.0     # [x,y,z]
radius2       = 35.0
fixatom       = YES
fix_layer     = 1.0
```

CHAPTER

TWELVE

SELECTION SECTION

12.1 Atom selection

This section is used to select atoms, and define them as a group. The user can select atoms accorging to their name, index, residue number, segment name, and so on. The selected group index is used in other sections. For example, restraint potential can be applied on the group selected in this section, and the force constant of the potential is specified in the **[RESTRAINTS]** section. **[SELECTION]** section is also used in the GENESIS analysis tools to specify the atoms to be analyzed.

---

group*N* *expression*

The user defines selected atoms as "group1", "group2", . . . , and "group:math:*N*". Here, *N* must be a positive integer ($N \geq 1$). The user selects atoms by using keywords and operators with a certain syntax (see table below). Note that in the table mname (or moleculename, molname) is a molecule name defined by **mol_name**.

mole_name*N* *molecule starting-residue ending-residue*

The user defines a molecule by specifying its segment name, first and last residue numbers, and residue name. `N` must be a positive integer ($N \geq 1$). The syntax for the residue selection is as follows:

```
[segment name]:[residue number]:[residue name]
```

For details, see the example below.

Table. Available keywords and operators in `group`.

expression	meaning	example	other available expression
an:*name*	atom name	an:CA	atomname, atom_name
ai:*number[-[number]]*	atom index	ai:1-5	atomindex, atomidx
atno:*number[-[number]]*	atom number	atno:6	atomno
rnam:*name*	residue name	rnam:GLY	residuename, resname
rno:*number[-[number]]*	residue number	rno:1-5	residueno, resno
mname:*name*	molecule name	mname:molA	moleculename, molname
segid:*ID*	segment index	segid:PROA	segmentid, sid
hydrogen	hydrogen atoms		hydrogenatom
heavy	heavy atoms		heavyatom
all	all atoms		*
and	conjunction		&
or	logical add		|
not	negation		!
()	assemble		

Table. Available keywords and operators in `group` (continued).

expression	meaning	example	other available expression
X around: *r*	atoms around *r* Åof *X*	see below	around_atoms
X around_res: *r*	residues around *r* Åof *X*	see below	around_residues
X around_mol: *r*	molecules around *r* Åof *X*	see below	around_molecules

Note: `ai` and `atno` are slightly different. `ai` indicates the atom index which is sequentially renumbered over all atoms in the system. On the other hand, `atno` is the index of atoms in the PDB file. Atom index in PDB file (column 2) does not always start from 1, nor is numbered sequentially. In such cases, `atno` is useful to select atoms, although it is a very rare case.

Note: Atoms that are within a distance of a given atom (X) can be selected by `around`. Note that the coordinates in `reffile` is used to judge the distance. If `reffile` is not present, those in input files (`pdbfile`, `crdfile`, etc.) are used instead. Coordinates in `rstfile` are never used.

12.2 Examples

Select atoms based on their atom name, residue name, or residue number:

```
[SELECTION]
group1     = resno:1-60 and an:CA
group2     = (segid:PROA and not hydrogen) | an:CA
mole_name1 = molA PROA:1:TYR PROA:5:MET
group3     = mname:molA and (an:CA or an:C or an:O or an:N)
```

Select atoms around an atom X. In the following examples, X = atom number 100.

```
[SELECTION]
group1     = atno:100 around:10.0
group2     = atno:100 around_res:10.0
group3     = atno:100 around_mol:10.0
group4     = atno:100 around_mol:10.0 or atno:100
```

In group1, atoms around 10.0 Åof X are selected. Group 2 selects residues around 10.0 Åof X, i.e., if the distance between X and any one of atoms in a residue is less than 10.0 Å, all atoms of the residue are selected. Group 3 is the same as group 2, but for a molecule. Note that these commands do NOT select X itself. In order to include X in the selection, add “or atno:100”, as in group 4.

Select atoms around multiple atoms.

```
[SELECTION]
group1     = atno:100-101 around:10.0
group2     = (sid:PROT around_res:10.0) and rnam:TIP3
group3     = (rno:1 around:10.0) or rno:1
```

Group 1 selects atoms around 10.0 Åof atom 100 *or* 101. Note that it is NOT “100 *and* 101” nor a center of 100 and 101. Group 2 is an example to select water molecules around a protein (segname PROT). Group 3 selects not only the atoms around residue1 but also the atoms of residue1.

CHAPTER

THIRTEEN

RESTRAINTS SECTION

13.1 Restraint potential

[RESTRAINTS] section contains keywords to define external restraint functions. The restraint functions are applied to the selected atom groups in **[SELECTION]** section to restrict the motions of those atoms.

The potential energy of a restaint can be written as:

$$U(x) = k\ (x - x_0)^n$$

where x is a variable (see bellow), x_0 is a reference value, k is a force constant, and n is an exponent factor.

13.1.1 General keywords

nfunctions *Integer*

> **Default: 0**
>
> Number of restraint functions.

function*N* *POSI / DIST[MASS] / ANGLE[MASS] / DIHED[MASS] / RMSD[MASS] / PC[MASS] / EM*

> **Default: N/A**
>
> Type of restraint.
>
> - **POSI**: positional restraint. The reference coordinates are given by **reffile**, **ambreffile**, or **groreffile** in **[INPUT]**. (see *Input section*)
> - **DIST[MASS]**: distance restraint.
> - **ANGLE[MASS]**: angle restraint.
> - **DIHED[MASS]**: dihedral angle restraint.
> - **RMSD[MASS]**: RMSD restraint. *MASS* means mass-weighted RMSD. Translational and rotational fitting to the reference coordiate are done before calculating RMSD. The reference coodinate is specified in the same manner as *POSI*.
>
>   *Important Notice (1.1.5 or later)* Structural fitting method can be defined in **[FITTING]** section (*Fitting section*) on 1.1.5 or later. Users of GENESIS 1.1.4 or before

should pay special attention on the fitting scheme. In versions 1.1.4 or before, translational and rotational fittings were automatically applied for the atoms concerning RMSD restraint. (same as the current default setting, *fitting_method* = *TR+ROT*)

- **PC[MASS]**: principal component constraint. This option requires *modefile* in the *Input section*.
- **EM**: cryo-EM flexible fitting (see *Experiments section*)

DIST, *ANGLE*, *DIHED* impose restraint on distance/angle/dihedral defined by the selected groups. See `select_indexN` and examples below for the specification. *MASS* indicates that the force is applied to the center of mass of the selected group. When *MASS* is omitted, the force is applied to the geometric center of the coordinates. *MASS* keyword does nothing for groups consist of a single particle.

In **SPDYN**, *POSI* and *RMSD[MASS]* restraints are mutually exclusive; you can use either one or none of them. Two different *POSI* restraints might not be applied simultaneously, either.

Notice: POSI, PC, and RMSD restraints can be influenced by the removal of translational/rotational momentum. See also the notices in the stoptr_period parameter in the [DYNAMICS] section.

constant*N* *Real*

Default: 0.0 (unit: depend on the restraint type)

Force constant of a restraint function. The unit depends on the type of restraint. Namely, $\mathrm{kcal/mol/Å}^n$ is used in the case of *DIST* and *RMSD*, while $\mathrm{kcal/mol/rad}^n$ in the case of *ANGLE* and *DIHED*, where n is **exponent*N*** specified in this section.

reference*N* *Real*

Default: 0.0 (unit: depend on the restraint type)

Reference value of a restraint function. For the positional restraint, the value is ignored. The unit depends the type of restraint. Namely, Åis used in the case of *DIST*, while degree (NOT radian) is used in the case of *ANGLE* and *DIHED*.

select_index*N* *Integer*

Default: N/A

Index of an atom group, to which restraint potentials are applied. The index must be defined in **[SELECTION]** (see *Selection section*). For example, if you specify `select_index1 = 1`, this restraint function is applied for `group1` in the **[SELECTION]**.

Number of groups required depends on the type of the restraint function.

- *POSI/RMSD[MASS]*: 1
- *DIST[MASS]*: $2m$, where $m = 1, 2, ...$
- *ANGLE[MASS]*: 3
- *DIHED[MASS]*: 4
- *PC[MASS]*: ≥ 1

A group can contain more than single atom. Suppose we have the following input.

```
[SELECTION]
group1        = ai:1-10
group2        = ai:11-20

[RESTRAINTS]
nfunctions    = 1
function1     = DIST
constant1     = 3.0
reference1    = 10.0
select_index1 = 1 2
```

In this case, the distance restraint is applied for the distance between geometric centers of group1 and group2. The calculated force is then scattered to each atom. If *DISTMASS* is given instead of *DIST*, mass centers (mass-weighted average position) are used instead of geometric centers (not mass-weighted average position).

In the case of *DIST[MASS]* restraint with more than 2 groups specified (i.e. $2m$ with $m \geq 2$), the sum of m distances will be restrained. See `exponent_dist` and `weight_dist` parameters for this distance summation. However, this scheme might not be useful for the standard cases.

directionN *ALL / X / Y / Z*

Default : ALL

Direction of the *POSI* restraint. If *X* or *Y* or *Z* is specified, restraints along the other two axes are not applied.

exponentN *Integer*

Default : 2

Exponent factor of the restraint function. The default is the harmonic. This parameter does not work for *POSI* and *RMSD[MASS]* restraints in **SPDYN**, where the default value, 2, is always used.

exponent_distN *Integer* (**DIST[MASS]** only)

Default : 1

Exponent factor used in the distance sum calculations. The sum of distances is expressed as: $r_{\mathrm{sum}} = \sum_m w|r_m|^n$, where $(1 \leq m \leq \mathrm{num\ groups}/2)$, n is **exponent_distN**, and w is **weight_distN**.

weight_distN *Real* (**DIST[MASS]** only)

Default : 1.0

Weight factor used in the distance sum calculations.

modeN *Integer*

Specifies the mode index which is used for the PC (principal component) restraint. For example, the 1st PC mode can be restrained by specifying `mode1=1`.

13.1.2 Pressure derived from restraints

Basically, the pressure calculated from the restraint potential is included in an internal pressure, which is kept constant during the simulation in the NPT ensemble. However, the pressure derived from positional and RMSD restraints are treated as an external pressure by default. Keywords **pressure_position** and **pressure_rmsd** are used to include those pressures in the internal pressure. If the simulation with POSI or RMSD restraint showed a strange behaviour (especially, when a strong force constant is applied), turn on these options.

pressure_position *YES / NO*

> **Default : NO**
>
> The virial terms from positional restraints are included in pressure evaluation.

pressure_rmsd *YES / NO*

> **Default : NO**
>
> The virial terms from RMSD restraints are included in pressure evaluation.

13.1.3 Advanced definition of restraints

Restraints can be also defined in an external input file (**localresfile**). In this case, number of local restraints must NOT be included in **nfunctions**. This option is availabe in **SPDYN** only. For details, see *Input section*.

13.1.4 Restraints in REUS simulations

If you employed a certain restraint term for REUS runs, *nreplica* of force constants and reference values must be given as a space-separated list. The above keywords except for **nfunctions**, **pressure_position**, and **pressure_rmsd**, must have a serial number, 'N', of the function ($N \geq 1$). This serial number is referred when selecting restraints in REUS runs. For details, see *REMD section*.

13.2 Examples

Example of **[RESTRAINTS]** section:

```
[RESTRAINTS]
nfunctions    = 1
function1     = DIST
reference1    = 10.0
constant1     = 2.0
select_index1 = 1 2        # group1 and group2 in [SELECTION]
```

Example of multiple restraints:

```
[RESTRAINTS]
nfunctions    = 2

function1     = DIST
constant1     = 2.0
```

(continues on next page)

(continued from previous page)

```
reference1    = 10.0        # in angstrom
select_index1 = 1 2

function2     = DIHED
constant2     = 3.0
reference2    = 120.0       # in degrees
select_index2 = 3 4 5 6
```

CHAPTER

FOURTEEN

FITTING SECTION

14.1 Structure fitting

(In GENESIS 1.1.5 or later only) Keywords in **[FITTING]** section define a structure superimposition scheme, which is often employed in targeted MD, steered MD, or String method (see *RPATH section*) with positional restraint. In the String method, the reference coordinate for fitting is given by **fitfile** in the **[INPUT]** section. Otherwise (MD, MIN, REMD), the reference coordinate is given by **reffile**, **ambreffile**, or **groreffile** in the **[INPUT]** section. Note that this section is not related to the cryo-EM flexible fitting (see *Experiments section*)

---

fitting_method *NO / TR+ROT / XYTR+ZROT*

Default: TR+ROT

Type of fitting method.

- NO: No fitting routine is applied
- TR+ROT: Remove both of translation and rotation
- XYTR+ZROT: Remove translation in XY-plane and rotation along the Z-axis

fitting_atom: *Integer*

Default: N/A

Index of an atom group which is to be fitted to the reference structure. In RMSD restraints, Steered MD, or Targeted MD, this should be identical to the group where the restraint potential is applied. The index must be defined in **[SELECTION]** (see *Selection section*). For example, if you specify `fitting_atom = 1`, the reference atoms are members of `group1` in the **[SELECTION]**.

mass_weight: *YES / NO*

Default: NO

If the parameter is set to *YES*, mass-weighted fitting is employed. This parameter should be *YES* for RMSDCOM/PCCOM restraints and should be *NO* for RMSD/PC restraints. Please make sure that this parameter is correctly specified when you perform RMSD/RMSDCOM/PC/PCOM type of calculations. In the String method, mass-weighted superimposition is not supported.

force_no_fitting: *YES / NO*

Default: NO

This parameter must not be changed for standard MD runs. If the parameter is set to YES and fitting_method is set to NO, the fitting routine is turned off. Translational and rotational fittings are usually required to calculate correct RMSD values. So GENESIS simulators (ATDYN and SPDYN) do not allow *fitting_method = NO* for simulations involving RMSD calculation (targeted/steered MD, for example). But such fitting is not desirable when generating initial structure set for the String method using Cartesian coordinate as CV (see *RPATH section*). Actually, *fitting_method = NO* was implemented just for this specific purpose. If you are really want to turn off fittings of RMSD calculation for preparation of initial structure set for String method, please specify *fitting_method = NO* and *force_no_fitting = YES*.

14.2 Examples

Example of **[FITTING]** section

```
[FITTING]
fitting_method = TR+ROT
fitting_atom   =  1
mass_weight    =  NO
```

CHAPTER FIFTEEN

REMD SECTION

15.1 Replica-exchange molecular-dynamics simulation (REMD)

In the **[REMD]** section, the users can specify keywords for Replica-Exchange Molecular Dynamics (REMD) simulation. REMD method is one of the enhanced conformational sampling methods used for systems with rugged free-energy landscapes. The original temperature-exchange method (T-REMD) is one of the most widely used methods in biomolecules' simulations [58] [59]. Here, replicas (or copies) of the original system are prepared, and different temperatures are assigned to each replica. Each replica runs in a canonical (NVT) or isobaric-isothermal (NPT) ensemble, and the temperatures are periodically exchanged between the neighboring replicas during a simulation. Exchanging temperature enforces a random walk in temperature space, allowing the system overcoming energy barriers and sampling much wider conformational space.

In REMD methods, the transition probability of the replica exchange process is given by the usual Metropolis criterion,

$$w(X \rightarrow X') = \min(1, \frac{P(X')}{P(X)}) = \min(1, \exp(-\Delta)).$$

In the T-REMD method, we have

$$\Delta = (\beta_m - \beta_n)\left\{E(q^{[j]}) - E(q^{[i]})\right\},$$

where E is the potential energy, q is the position of atoms, β is the inverse temperature defined by $\beta = 1/k_B T$, i and j are the replica indexes, and m and n are the parameter indexes. After the replica exchange, atomic momenta are rescaled as follows:

$$p^{[i]\prime} = \sqrt{\frac{T_n}{T_m}} p^{[i]}, \qquad p^{[j]\prime} = \sqrt{\frac{T_m}{T_n}} p^{[j]},$$

where T is the temperature and p is the momenta of atoms.

The transition probability should be independent of the algorithms used: i.e. constant temperature and constant pressure algorithms. On the other hand, the momenta-rescaling scheme depends on the algorithm used in the simulation. If thermostat and barostat momenta are included in the equations of motion, these variables should be also rescaled after replica exchange [60] [61]. In GENESIS, barostat momentum is rescaled in the case of T-REMD with Langevin or Bussi method in NPT, NPAT, and NPgT ensembles. For the other cases, only atomic momenta are rescaled.

In GENESIS, not only Temperature REMD but also pressure REMD [62], surface-tension REMD [63], REUS (or Hamiltonian REMD) [64] [65], replica exchange with solute tempering (REST) [66] [67], and their multi-dimensional combinations are available in both **ATDYN** and **SPDYN**. Basically, these methods can be employed in the NVT, NPT, NPAT, NPgT ensembles, except for the surface-tension REMD, which is only used in the NPgT ensemble. REMD simulations in GENESIS require an MPI environment. At least one MPI process must be assigned to one replica. For example, when the user wants to employ 32 replicas, $32n$ MPI processes are required.

In the following parameters excluding *dimension*, *exchange_period*, and *iseed*, the last character 'N' must be replaced with a positive integer number (i.e. $N \geq 1$), which defines the index of replica dimension. For example, *type1*, *nreplica1* are the replica type and number of replicas for the first dimension, respectively. For details, see the examples below.

---

dimension *Integer*

Default: 1

Number of dimensions (i.e. number of parameter types to be exchanged)

type*N* *TEMPERATURE / PRESSURE / GAMMA / RESTRAINT / REST*

Default: TEMPERATURE

Type of parameter to be exchanged in the N-th dimension

- **TEMPERATURE**: Temperature REMD [58]
- **PRESSURE**: Pressure REMD [62]
- **GAMMA**: Surface-tension REMD [63]
- **RESTRAINT**: REUS (or Hamiltonian REMD) [64] [65]
- **REST**: replica exchange with solute tempering (REST2 or gREST) [66] [67], which is totally different from the original version of REST [68]. Currently, only AMBER and CHARMM force fields are supported.
- **ALCHEMY**: FEP/λ-REMD [69]

nreplica*N* *Integer*

Default: 0

Number of replicas (or parameters) in the N-th dimension

parameters*N* *Real*

Default: N/A

List of parameters for each replica in the N-th dimension. Parameters must be given as a space-separated list, and the total number of parameters must be equal to **nreplicaN**. In case of REUS (type = RESTRAINT), parameters must be specified in **[RESTRAINTS]** section (see the sample below). In case of gREST (type = REST), these parameters are considered as temperature of solute region. Note that the order of the parameters in this list must NOT be changed before and after the restart run, even if the parameters are exchanged during the REMD simulation.

exchange_period *Integer*

Default: 100

Frequency of the parameter exchange attempt. If "exchange_period = 0" is specified, REMD simulation is carried out without parameter exchange, which is useful to equilibrate the system in a condition assinged to each replica before performing the production run.

cyclic_params*N* *YES / NO*

Default: NO

Turn on or off the periodicity of the parameters in the N-th dimension. If "cyclic_paramsN = YES" is specified, the first and last parameters are considered as neighbouring parameters. This option can be applicabe to all parameter types. Basically, this is useful in the case of REUS in dihedral angle space, since the dihedral angle has a periodicity.

iseed *Integer*

Default: 3141592

Random number seed in the replica exchange scheme. If this is not specified explicitly, iseed is taken over from the restart file.

Note: In the [ENSEMBLE] section, there is also a parameter "temperature". In the T-REMD simulation, this temperature is ignored, even if it is specified explicitly. Similarly, pressure and gamma in the [ENSEMBLE] section are ignored in the P-REMD and surface-tension REMD simulations, respectively.

Note: When multi-dimensional REMD is carried out, parameters are exchanged alternatively. For example, in TP-REMD (type1 = TEMPERATURE and type2 = PRESSURE), there is a temperature exchange first, followed by a pressure exchange. This is repeated during the simulations.

15.2 Replica-exchange umbrella-sampling (REUS)

rest_function*N* (for **REUS** only)

Index of the restraint function to be used in the REUS simulation. The detailed parameters in the restraint function (e.g., force constant and reference) are defined in the **[RESTRAINTS]** section (see *Restraints section*). Note that the order of the parameters in the **[RESTRAINTS]** section must NOT be changed before and after the restart run, even if the parameters are exchanged during the REUS simulation.

GENESIS supports not only on-grid but also off-grid schemes. In the off-grid REUS, multiple restraints are merged into a single reaction coordinate (see example below). Those restraints are defined in **[RESTRAINTS]** section, where the number of parameters (const and reference) must be equal to *nreplicaN*. Note that this kinds of combined axis can be used only for restraints, other types (such as combined tempreature-pressure or temperature-restraint coordinate) are not available currently.

Note: Positional restraint is not available for REUS. In **SPDYN**, PCA restraint is not available for REUS. The control file format was completely changed after verion 1.1.0, since the off-grid REUS scheme was introduced. When the users use the old control file, please be careful.

15.3 Replica-exchange with solute-tempering (gREST)

select_index*N Integer*

Default: N/A

Index of an atom group. The selected atoms are considered as "solute" in gREST. The index must be defined in **[SELECTION]** (see *Selection section*).

param_type*N ALL / BOND / ANGLE / UREY / DIHEDRAL / IMPROPER / CMAP / CHARGE / LJ*

Default: ALL

Solute energy terms for gREST [67] simulations. Energy terms selected by this parameter in the solute atom group (defined by *select_indexN*) are considered as "solute" (scaled according to solute temperature) in gREST. Other terms are considered as "solvent" (kept intact). Solute-solvent terms are automatically determined from the solute selection. You can specify multiple terms (see examples). The parameter names are case-insensitive as follows:

- **ALL**: all the available energy terms.
- **BOND**: (aliases: **B**, **BONDS**): 1-2 bonding terms.
- **ANGLE**: (aliases: **A**, **ANGLES**): 1-2-3 angle terms.
- **UREY**: (aliases: **U**, **UREYS**): Urey-Bradley terms.
- **DIHEDRAL**: (aliases: **D**, **DIHEDRALS**): 1-2-3-4 dihedral terms.
- **IMPROPER**: (aliases: **I**, **IMPROPERS**): improper torsion terms.
- **CMAP**: (aliases: **CM**, **CMAPS**): CMAP terms.
- **CHARGE**: (aliases: **C**, **CHARGES**): coulombic interaction terms.
- **LJ**: (aliases: **L**, **LJS**): Lennard-Jones interaction terms.

Note: Note that restraint energy terms defined in **[RESTRAINTS]** cannot be treated as solute terms. They never be affected by gREST solute temperatures. In SPDYN, water atoms cannot be specified as "solute" now. This limitation will be removed in the future version.

Note: When the coulombic interaction terms are considered as the solute, the solute region should have a net charge of 0 for an adequate PME calculation.

15.4 Examples

Basically, REMD simulations in **GENESIS** can be carried out by just adding the **[REMD]** section in the control file of a normal MD simulation. For details, see the online Tutorial (https://www.r-ccs.riken.jp/labs/cbrt/tutorials2019/).

15.4.1 T-REMD

If the users want to carry out T-REMD simulations with 4 replicas in the NVT ensemble, where each replica has the temperature 298.15, 311.79, 321.18, or 330.82 K, and replica exchange is attempted every 1000 steps, the following section is added to the control file of a normal MD simulation in the NVT ensemble:

```
[REMD]
dimension        = 1
exchange_period  = 1000
type1            = TEMPERATURE
nreplica1        = 4
parameters1      = 298.15 311.79 321.18 330.82
```

As for the T-REMD simulation in the NPT ensemble, the users add this section to the control file of a normal MD simulation in the NPT ensemble. The REMD temperature generator (http://folding.bmc.uu.se/remd/) is a useful tool to set the target temperature of each replica.

15.4.2 Two-dimensional REMD (T-REMD/REUS)

The following is an example of two-dimensinal REMD, where temperature and restraint are exchanged alternatively, The 1st dimension is T-REMD with 8 parameters, and 2nd dimension is REUS in distance space with 4 parameters. In total, 8 x 4 = 32 replicas are used:

```
[REMD]
dimension        = 2
exchange_period  = 1000
type1            = TEMPERATURE
nreplica1        = 8
parameters1      = 298.15 311.79 321.18 330.82 340.70 350.83 361.23 371.89
type2            = RESTRAINT
nreplica2        = 4
rest_function2   = 1

[SELECTION]
group1           = ai:25
group2           = ai:392

[RESTRAINTS]
nfunctions       = 1
function1        = DIST
constant1        =  2.0   2.0   2.0   2.0
reference1       = 10.0  10.5  11.0  11.5
select_index1    = 1 2
```

These sections are added to the control file of a normal MD simulation.

15.4.3 Off-grid REUS

Example of off-grid REUS (merge two restraints into single reaction coordinate), where distance and dihedral restraints are merged into single reaction coordinate. First values of restraints ((2.0,10.0) for distance, (10,-40) for dihedral) will be used for the first replica, the fourth parameters ((2.0,11.5) for distance, (10,-10) for dihedral) will be used for the fourth replica:

```
[REMD]
dimension        = 1
exchange_period = 1000
type1            = RESTRAINT            # REUS
nreplica1        = 4
rest_function1   = 1 2                  # off-grid REUS

[SELECTION]
group1           = ai:25
group2           = ai:392
group3           = ai:72
group4           = ai:73
group5           = ai:74
group6           = ai:75

[RESTRAINTS]
nfunctions       = 2

function1        = DIST
constant1        =  2.0  2.0  2.0  2.0  # num of values must be nreplica1
reference1       = 10.0 10.5 11.0 11.5
select_index1    = 1 2

function2        = DIHED
constant2        =   10   10   10   10  # num of values must be nreplica1
reference2       =  -40  -30  -20  -10
select_index2    = 3 4 5 6
```

15.4.4 gREST

In this example, the dihedral, CMAP, and LJ energy terms in the selected atom groups are treated as "solute".

```
[REMD]
dimension        = 1
exchange_period = 1000
type1            = REST
nreplica1        = 4
parameters1      = 300.0 310.0 320.0 330.0  # solute temperatures
param_type1      = D CM L                   # dihedral, CMAP, and LJ
select_index1    = 1

[SELECTION]
group1           = ai:1-313
```

T-REMD in the two-dimensional REMD (T-REMD/REUS) may be replaced with gREST (gREST/REUS [70]) to reduce the required number of replicas.

```
[REMD]
dimension        = 2
exchange_period = 1000
type1            = REST
nreplica1        = 4
```

(continues on next page)

(continued from previous page)

```
parameters1      = 300.0 310.0 320.0 330.0  # solute temperatures
param_type1      = D CM L                   # dihedral, CMAP, and LJ
select_index1    = 3
type2            = RESTRAINT
nreplica2        = 4
rest_function2   = 1

[SELECTION]
group1           = ai:25
group2           = ai:392
group3           = ai:1-313

[RESTRAINTS]
nfunctions       = 1
function1        = DIST
constant1        =  2.0   2.0   2.0   2.0
reference1       = 10.0  10.5  11.0  11.5
select_index1    = 1 2
```

CHAPTER

SIXTEEN

RPATH SECTION

16.1 Reaction Path Search

In the **[RPATH]** section, users can specify keywords for finding the reaction path. The path search is carried out in two modes: the minimum energy path (MEP) and the minimum free-energy path (MFEP). The former searches for the energetically most favorable pathway on the potential energy surface (PES), while the latter does the same on the free-energy surface (FES). The MEP search is used to find relatively fast processes such as chemical reactions (very likely along with QM/MM), in which the environment can be regarded more or less rigid. On the other hand, the MFEP reveals large-scale conformational changes of biomolecules by searching the path on a FES, where fast molecular motions are averaged out. In both cases, the path is represented by a chain-of-replica, which evolves on the energy surface so as to minimize the forces in the transverse direction.

---

rpathmode *MFEP/MEP*

Default: MFEP

Specify **MFEP** or **MEP** to invoke the MFEP or MEP search.

16.2 Minimum Free-Energy Path (MFEP) Search

The MFEP search is invoked by specifying **rpathmode = MFEP**. The path search is carried out by the string method, which is a powerful sampling technique to find a path connecting two stable conformational states. This method is widely used for investigating large-scale conformational changes of biomolecules where time-scale of the transitions are not reachable in brute-force simulations.

There are three major algorithms in the string method: the mean forces string method [71], the on-the-fly string method [72], and the string method of swarms of trajectories [73]. Among these algorithgms, the mean forces string method is available in **ATDYN** and **SPDYN** [74].

In the mean-forces string method, the pathway is represented by discretized points (called images) in the collective variable (CV) space. The current GENESIS supports distances, angles, dihedrals, Cartesian coordinates, and principal components for CVs (note that different types of CVs cannot be mixed in GENESIS. For example, users cannot mix distance and angle).

In the calculation, each image is assigned to each replica, and a replica samples mean forces and an average metric tensor around its own image by short MD simulation (ps to ns length) with restraints.

The restraints are imposed using the image coordinates as their reference values. After the short simulation , each image is evolved according to the mean force and metric tensor. Then, smoothing and re-parametrization of images are performed and go to the next cycle.

Image coordinates are written in rpath files (**rpathfile** keyword) which user can specify in **[OUTPUT]** section. This file provides the trajectory of image coordinates. Columns correspond to CVs and rows are time steps. These values are written at the same timing with **dcdfile** (specified by **crdout_period** in **[DYNAMICS]** section).

For the string method calculation, an initial pathway in the CV space and atomistic coordinates around the pathway are required. For preparing these, targeted or steered MD methods are recommended.

---

nreplica *Integer*

> **Default: 1**
>
> Number of replicas (images) for representing the pathway.

rpath_period *Integer*

> **Default: 0**
>
> Time-step period during which the mean-forces acting on the images are evaluated. After evaluating the mean-forces, the images are updated according to the mean-forces, then go to the next cycle. If **rpath_period = 0**, images are not updated. This option is used for equilibration or umbrella sampling around the pathway.

delta *Real*

> **Default: 0.0**
>
> Step-size for steepest descent update of images.

smooth *Real*

> **Default: 0.0**
>
> Smoothing parameter which controls the aggressiveness of the smoothing. Values from 0.0 to 0.1 are recommended, where “smooth = 0.0” means no-smoothing

rest_function *List of Integers*

> **Default: N/A**
>
> List of restraint function indices defined in **[RESTRAINTS]** section (see *Restraints section*). Specified restraints are defined as CVs, and *nreplica* images (replicas) are created, where a set of corresponding restraint reference values is assigned to each image. Force constants in **[RESTRAINTS]** are also used for evaluation of mean-forces.

fix_terminal *YES / NO*

> **Default: NO**
>
> If **fix_terminal = YES** is specified, the two terminal images are always fixed and not updated. This is useful if the terminal images correspond to crystal structures and users do not want to move them.

use_restart *YES / NO*

Default : YES

Restart file generated by the string method calculation includes the last snapshot of images. If **use_restart = YES** is specified, the reference values in **[RESTRAINTS]** will be overwritten by the values in the restart file. Note that force constants are not overwritten.

Note: The following options are needed in the **[FITTING]** section when Cartesian coordinates are used for CVs.

fitting_method *TR+ROT / XYTR+ZROT / NO*

If this keyword is specified, rot-translational elements are removed from the mean-force estimation by fitting instaneous structures to the reference coordinates given by **fitfile**.

fitting_atom *List of Integers*

The user can specify index of an atom group which are fitted to the reference structure. Usually, the same atoms as CVs are selected.

16.3 Minimum Energy Path (MEP) Search

The MEP search is available only in **ATDYN**.

The calculation is invoked by specifying **rpathmode = MEP**. Cartesian coordinates of atoms selected via **mepatm_select_index** (denoted MEP atoms) are employed for the path search. Currently, other coordinates can not be used as CVs. Starting from a set of images along an initial path, e.g., a linear interpolation between the reactant and product, the coordinates of the surrounding atoms are first energy minimized with MEP atoms held fixed. (The [MINIMIZE] section is thus required in the input as well.) Then, the forces acting on MEP atoms are evaluated for each image, and the images are evolved so as to minimize the forces in the transverse direction. The process is repeated either until the convergence threshold is met [variation in the energy (**tol_energy**) and the path length (**tol_path**)], or until the number of iterations reaches the maximum number (**ncycle**).

Two search algorithms are implemented in GENESIS: the string method [75] and the nudged elastic band (NEB) method [76]. In the NEB method, the spring constant (**k_spring**) should be carefully chosen; in this sense, NEB is still experimental. The string method is more stable and thus recommended by default, although the convergence often slower than in NEB.

---

mepatm_select_index *Integer*

Index of a group of atoms which is treated as MEP atoms. The index must be defined in **[SELECTION]** (see *Selection section*).

ncycle *Integer*

Default: 1000

Maximum number of cycle.

nreplica *Integer*

Default: 1

Number of replicas (images) for representing the pathway.

Note: If MPI processes are larger than **nreplica**, the MPI processes must be a multiple of **nreplica**. For example, if **nreplica = 16**, MPI processes must be 16, 32, 48, etc. If MPI processes are smaller than **nreplica**, the MPI processes must be a divisor of **nreplica**. For example, the calculation with **nreplica = 16** can be performed using 1, 2, 4, and 8 MPI processes.

eneout_period *Integer*

Default : 10

Frequency of the output of the energy profile and path length to the standard output.

crdout_period *Integer*

Default : 0

Frequency of coordinates outputs. Note that coordinate outputs are turned off for the minimization (**crdout_period** in the [MINIMIZE] section).

rstout_period *Integer*

Default : 0

Frequency of restart file updates. Note that the updates are turned off for the minimization (**rstout_period** in the [MINIMIZE] section).

tol_energy *Real*

Default : 0.01 (unit : kcal/mol)

Tolerence of convergence for the energy.

tol_path *Real*

Default : 0.01 (unit : Å)

Tolerence of convergence for the path length.

massweightcoord *YES / NO*

Default : NO

Use mass weighted Cartesian.

method *STRING/NEB*

Default: STRING

Choose the algorithm of a MEP search.

Options for String.

delta *Real*

Default: 0.001

Step-size for steepest descent update of images.

Options for NEB.

k_spring *Real*

Default: 10.0 $\mathrm{kcal/mol/Å}^2$

Spring constant of the force that connects the images

ncorrection *Integer*

Default : 10

Number of corrections to build the inverse Hessian.

lbfgs_bnd *YES / NO*

Default : YES

Set a boundary to move atoms in each step of image update.

lbfgs_bnd_qmonly *YES / NO*

Default : NO

Set the boundary only to QM atoms.

lbfgs_bnd_maxmove *Real*

Default : 0.1 (unit : Å)

The maximum size of move in each step.

16.4 Examples

Example of alanine-tripeptide with 16 replicas (images). Two dihedral angles are specified as the collective variables.

```
[RPATH]
nreplica            = 16
rpath_period        = 1000
delta               = 0.02
smooth              = 0.0
rest_function       = 1 2

[SELECTION]
group1        = atomindex:15
group2        = atomindex:17
group3        = atomindex:19
group4        = atomindex:25
group5        = atomindex:27

[RESTRAINTS]
nfunctions    = 2

function1     = DIHED
constant1     = 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 \
                100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0
reference1    = -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 \
                -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 -40.0 -40.0
select_index1 = 1 2 3 4   # PHI
```

(continues on next page)

(continued from previous page)

```
function2      = DIHED
constant2      = 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0 \
                 100.0 100.0 100.0 100.0 100.0 100.0 100.0 100.0
reference2     = -45.0 -33.0 -21.0 -9.0 3.0 15.0 27.0 39.0 \
                  51.0  63.0  75.0 87.0 99.0 111.0 123.0 135.0
select_index2 = 2 3 4 5  # PSI
```

Here is another example of Cartesian coordiante CVs for the same alanine-tripeptide.

```
[INPUT]
... skip ...
rstfile = ../eq/{}.rst
reffile = {}.pdb
fitfile = fit.pdb

[RPATH]
nreplica          = 16
rpath_period      = 1000
delta             = 0.001
smooth            = 0.00
rest_function     = 1
fix_terminal      = NO

[FITTING]
fitting_method = TR+ROT
fitting_atom   = 1

[SELECTION]
group1         = ai:15 or ai:17 or ai:19 or ai:25 or ai:27

[RESTRAINTS]
nfunctions     = 1

function1      = POSI
constant1      = 10.0 10.0 10.0 10.0 \
                 10.0 10.0 10.0 10.0 \
                 10.0 10.0 10.0 10.0 \
                 10.0 10.0 10.0 10.0
select_index1 = 1
```

Here is an example of a MEP search along with QM/MM

```
[INPUT]
topfile = toppar/top_all36_prot.rtf, ...
parfile = toppar/par_all36_prot.prm, ...
psffile = prot.psf                # protein structure file
reffile = prot.pdb                # PDB file
pdbfile = initial/initial{}.pdb  # initial path

[OUTPUT]
dcdfile = mep_{}.dcd        # coordinates
logfile = mep_{}.log        # log files
rstfile = mep_{}.rst        # restart file
rpathfile  = mep_{}.rpath   # rpath file
```

(continues on next page)

(continued from previous page)

```
[ENERGY]
forcefield       = CHARMM    # CHARMM force field
... skip ...

[MINIMIZE]
method             = LBFGS  # MIN using L-BFGS
nsteps             = 100    # max. number of steps
eneout_period      = 5      # energy output period
fixatm_select_index = 2     # fix the outer layer
macro              = yes    # macro/micro iteration

[RPATH]
rpathmode          = MEP    # MEP search
method             = STRING # String method
delta              = 0.0005 # step size
ncycle             = 200    # max. number of cycle
nreplica           = 16     # number of replica
eneout_period      = 1      # frequency of the energy output
crdout_period      = 1      # frequency of the coordinate output
rstout_period      = 1      # frequency of the restart update
fix_terminal       = no     # fix the terminal
massWeightCoord    = no     # mass-weighted Cartesian
mepatm_select_index = 1     # selection of the MEP atoms

[BOUNDARY]
type          = NOBC

[QMMM]
qmtyp             = gaussian
qmatm_select_index = 1
... skip ...

[SELECTION]
group1 = sid:DHA or (sid:TIMA and (rno:95 or rno:165) and \
         not (an:CA | an:C | an:O | an:N | an:HN | an:HA))
group2 = not (sid:DHA  or sid:DHA  around_res:6.0)
```

CHAPTER

SEVENTEEN

GAMD SECTION

17.1 Gaussian accelerated Molecular Dynamics

In the **[GAMD]** section, the users can specify keywords for Gaussian accelerated Molecular Dynamics (GaMD) simulation. The GaMD method [77][78] accelerates the conformational sampling of biomolecules by adding a harmonic boost potential to smooth their potential energy surface. GaMD has the advantage that reaction coordinates do not need to be predefined, thus setting up the system for the simulation is rather easy. The use of the harmonic boost potential allows to recover the unbiased free-energy changes through cumulant expansion to the second order, which resolves the practical reweighting problem in the original accelerated MD method.

GaMD was developed as a potential-biasing method for enhanced sampling. It accelerates the conformational sampling of a biomolecule by adding a non-negative boost potential to the system potential energy $U(\vec{x})$:

$$U'(\vec{x}) = U(\vec{x}) + \Delta U^{\mathrm{GaMD}}\left(U(\vec{x})\right),$$

where $\vec{x}$ is the configuration of the system, $U'(\vec{x})$ is the modified potential energy, and ΔU^{GaMD} is the boost potential depending only on $U(\vec{x})$.

In conventional accelerated MD [79][80][81], the average of the Boltzmann factors of the boost potential terms appears in the reweighting equation of the probability along the selected reaction coordinates, causing a large statistical error. In order to reduce the noise, GaMD uses a harmonic boost potential, which adopts a positive value only when the system potential is lower than an energy threshold E:

$$\Delta U^{\mathrm{GaMD}}\left(U(\vec{x})\right) = \begin{cases} \frac{1}{2}k\{E - U(\vec{x})\}^2 & (U(\vec{x}) < E) \\ 0 & (U(\vec{x}) \geq E) \end{cases},$$

where k is a harmonic force constant. $U'(\vec{x})$ should satisfy the following relationships [77][78]: $U'(\vec{x}_1) < U'(\vec{x}_2)$ and $U'(\vec{x}_2) - U'(\vec{x}_1) < U(\vec{x}_2) - U(\vec{x}_1)$ if $U(\vec{x}_1) < U(\vec{x}_2)$. To keep the relationships, the threshold energy needs to be set as:

$$U_{\mathrm{max}} \leq E \leq U_{\mathrm{min}} + \frac{1}{k},$$

where U_{max} and U_{min} are maximum and minimum energies of the system, respectively. To ensure accurate reweighting, the deviation of the potential must also satisfy the relation:

$$k(E - U_{\mathrm{ave}})\sigma_U \leq \sigma_0,$$

where $U_{\rm ave}$ and σ_U are the average and standard deviation of $U(\vec{x})$, respectively. σ_0 is a user-specified upper limit. k_0 is defined as $k_0 \equiv k(U_{\rm max} - U_{\rm min})$, then $0 < k_0 \leq 1$.

When E is set to the lower bound $U_{\rm max}$, k_0 is determined by

$$k_0 = \min\left(1, \frac{\sigma_0}{\sigma_U}\frac{U_{\rm max} - U_{\rm min}}{U_{\rm max} - U_{\rm ave}}\right)$$

When E is set to the upper bound $U_{\rm min} + 1/k$, k_0 is set to

$$k_0'' \equiv \left(1 - \frac{\sigma_0}{\sigma_U}\right)\frac{U_{\rm max} - U_{\rm min}}{U_{\rm ave} - U_{\rm min}}$$

if $0 < k_0'' < 1$, and k_0 is set to 1 otherwise.

The above parameters ($U_{\rm max}$, $U_{\rm min}$, $U_{\rm ave}$, and σ_U) are determined from short-time simulations a priori. When the distribution of the boost potential approaches Gaussian distribution, the cumulant expansion of the average of $\exp[\beta\Delta U^{\rm GaMD}]$ to the second order provides a good approximation for the free energy [82].

GaMD can be combined with REUS in such a way that each replica in REUS is accelerated by the GaMD boost potential:

$$\begin{aligned} U_i''(\vec{x}) &= U'(\vec{x}) + \Delta U_i^{\rm REUS}\left(\xi(\vec{x})\right) \\ &= U(\vec{x}) + \Delta U^{\rm GaMD}\left(U(\vec{x})\right) + \Delta U_i^{\rm REUS}\left(\xi(\vec{x})\right), \end{aligned}$$

where $U_i''(\vec{x})$ is the modified potential energy of replica i, $\Delta U_i^{\rm REUS}$ is the bias potential of REUS for replica i, and $\xi(\vec{x})$ is the collective variable of REUS. This method is referred to as Gaussian accelerated replica exchange umbrella sampling (GaREUS) [83]. The parameters in the GaMD boost potential are used in all replicas of GaREUS simulations. By using this combination, the simulated system in each replica becomes more flexible, or the energy barrier irrelevant to the collective variable is lowered, enhancing the sampling efficiency. When performing GaREUS simulations, the user must specify [REMD] section to use REUS and define a collective variable in the [SELECTION] and [RESTRAINTS] sections. Please check the example below.

---

gamd *YES / NO*

Default : NO

Enable the GaMD method.

boost *YES / NO*

Default : YES

Flag to apply GaMD boost to the system (). If *boost* = *NO*, boost is not applied but GaMD parameters are updated from the trajectory.

boost_type *DUAL / DIHEDRAL / POTENTIAL*

Default: DUAL

Type of boost.

- **DUAL**: Boost is applied on both the dihedral and total potential energies.

- **DIHEDRAL**: Boost is applied on only the dihedral energy.
- **POTENTIAL**: Boost is applied on only the total potential energy.

thresh_type *LOWER / HIGHER*

Default: LOWER

Type of threshold.

- **LOWER**: E is set to the lower bound $E = U_{\text{max}}$.
- **HIGHER**: E is set to its upper bound $E = U_{\text{min}} + 1/k$.

update_period *Integer*

Default: 0

Period of updating parameters in units of time step.

sigma0_pot *Real*

Default: 6.0 (unit: kcal/mol)

Upper limit of the standard deviation of the total potential boost (σ_0^{pot}) that allows for accurate reweighting.

pot_max *Real*

Default: -99999999.0 (unit: kcal/mol)

Maximum of the total potential energy of the system $U_{\text{max}}^{\text{pot}}$.

pot_min *Real*

Default: 99999999.0 (unit: kcal/mol)

Minimum of the total potential energy of the system $U_{\text{min}}^{\text{pot}}$.

pot_ave *Real*

Default: 0.0 (unit: kcal/mol)

Average of the total potential energy of the system $U_{\text{ave}}^{\text{pot}}$.

pot_dev *Real*

Default: 0.0 (unit: kcal/mol)

Standard deviation of the total potential energy of the system σ_U^{pot}.

sigma0_dih *Real*

Default: 6.0 (unit: kcal/mol)

Upper limit of the standard deviation of the dihedral boost (σ_0^{dih}) that allows for accurate reweighting.

dih_max *Real*

Default: -99999999.0 (unit: kcal/mol)

Maximum of the dihedral energy of the system $U_{\text{max}}^{\text{dih}}$.

dih_min *Real*

Default: 99999999.0 (unit: kcal/mol)

Minimum of the dihedral energy of the system $U^{\text{dih}}_{\text{min}}$.

dih_ave *Real*

Default: 0.0 (unit: kcal/mol)

Average of the dihedral energy of the system $U^{\text{dih}}_{\text{ave}}$.

dih_dev *Real*

Default: 0.0 (unit: kcal/mol)

Standard deviation of the dihedral energy of the system σ^{dih}_{U}.

17.2 Examples

Example of a GaMD simulation to determine initial parameters. To obtain the initial guess of the boost potential, (pot_max, pot_min, pot_ave, pot_dev, dih_max, dih_min, dih_ave, dih_dev) are calculated from a short simulation without boosting.

```
[GAMD]
gamd          = yes
boost         = no
boost_type    = DUAL
thresh_type   = LOWER
sigma0_pot    = 6.0
sigma0_dih    = 6.0
update_period = 50000
```

Example of a GaMD simulation updating parameters. The boost potential is updated every *update_period* during the simulation.

```
[GAMD]
gamd          = yes
boost         = yes
boost_type    = DUAL
thresh_type   = LOWER
sigma0_pot    = 6.0
sigma0_dih    = 6.0
update_period = 500
pot_max       = -20935.8104
pot_min       = -21452.3778
pot_ave       = -21183.9911
pot_dev       = 78.1207
dih_max       = 16.4039
dih_min       = 8.5882
dih_ave       = 11.0343
dih_dev       = 1.0699
```

Example of a GaMD simulation for production. In order to fix the parameters (pot_max, pot_min, pot_ave, pot_dev, dih_max, dih_min, dih_ave, dih_dev), *update_period* is set to 0.

```
[GAMD]
gamd          = yes
boost         = yes
boost_type    = DUAL
thresh_type   = LOWER
sigma0_pot    = 6.0
sigma0_dih    = 6.0
update_period = 0
pot_max       = -20669.2404
pot_min       = -21452.3778
pot_ave       = -20861.5224
pot_dev       = 48.9241
dih_max       = 23.2783
dih_min       = 8.5882
dih_ave       = 13.3806
dih_dev       = 1.7287
```

Example of a GaREUS simulation. The same GaMD parameters are applied in each replica of REUS. After the simulation, the two-step reweighting procedure using the multistate Bennett acceptance ratio method and the cumulant expansion for the exponential average is required to obtain the unbiased free-energy landscapes.

```
[REMD]
dimension         = 1
exchange_period   = 5000
type1             = RESTRAINT
nreplica1         = 4
rest_function1    = 1

[GAMD]
gamd          = yes
boost         = yes
boost_type    = DUAL
thresh_type   = LOWER
sigma0_pot    = 6.0
sigma0_dih    = 6.0
update_period = 0
pot_max       = -26491.7344
pot_min       = -27447.4316
pot_ave       = -26744.5742
pot_dev       = 52.5674
dih_max       = 135.8921
dih_min       = 91.2309
dih_ave       = 116.8572
dih_dev       = 3.6465

[SELECTION]
group1 = rno:1  and an:CA
group2 = rno:10 and an:CA

[RESTRAINTS]
nfunctions    = 1
function1     = DISTMASS
constant1     = 1.0 1.0 1.0 1.0
reference1    = 5.0 6.0 7.0 8.0
```

(continues on next page)

(continued from previous page)

```
select_index1 = 1 2
```

CHAPTER

EIGHTEEN

QMMM SECTION

18.1 Quantum mechanics/Molecular mechanics method (QM/MM)

QM/MM is available only in **ATDYN**.

The QM/MM method, first proposed in seminal papers by Warshel, Levitt, and Karplus [84] [85], is a multi-scale approach, which treats a partial region of interest (QM region) by quantum chemistry, and the surrounding environment (MM region) by force field. The method is useful, in particular, when the QM region involves an event that cannot be described by the standard force field; for example, chemical reactions, spectroscopy, etc.

In the QM/MM method, the potential energy of the system is written as,

$$V(\mathbf{R}_a, \mathbf{R}_m) = V^{\mathrm{QM}}(\mathbf{R}_a, \mathbf{R}_m) + V_{\mathrm{LJ}}^{\mathrm{QM-MM}}(\mathbf{R}_a, \mathbf{R}_m) + V^{\mathrm{MM}}(\mathbf{R}_m),$$

where $\mathbf{R}_a$ and $\mathbf{R}_m$ denote the position of atoms in QM and MM regions, respectively. $V_{\mathrm{LJ}}^{\mathrm{QM-MM}}$ and V^{MM} are the Lennard-Jones interaction between QM-MM atoms and the force field for MM atoms, respectively. The QM energy, V^{QM}, is written in terms of the electronic energy and the Coulomb interaction between nucleus-nucleus and nucleus-MM atoms,

$$V^{\mathrm{QM}}(\mathbf{R}_a, \mathbf{R}_m) = E_e(\mathbf{R}_a, \mathbf{R}_m) + \sum_{a>a'} \frac{Z_a Z_{a'}}{r_{aa'}} + \sum_{a,m} \frac{Z_a q_m}{r_{am}},$$

where Z_a and q_m are the charge of nucleus and MM atoms, respectively, and $r_{aa'}$ and r_{am} denote the distantce between nucleus and nucleus-MM atoms, respectively. The electronic energy is given by solving the Schrödinger equation for electrons,

$$\left[-\frac{1}{2}\sum_i \nabla_i^2 + \sum_{i>j} \frac{1}{r_{ij}} - \sum_{i,a} \frac{Z_a}{r_{ia}} - \sum_{i,m} \frac{q_m}{r_{im}} \right] |\Psi_e\rangle = E_e |\Psi_e\rangle,$$

where i, a, and m are indices for electrons, nucleus, and MM atoms, respectively, and r_{XY} denotes the distance between particle X and Y.

GENESIS does not have a function to solve the electronic Schrödinger equation, but is interfaced with external QM programs, which provide the QM energy, its derivatives, and other information. The interface is currently avaliable for Gaussian, Q-Chem, TeraChem, and DFTB+.

- Gaussian09/Gaussian16 (http://gaussian.com)
- Q-Chem (http://www.q-chem.com)
- TeraChem (http://www.petachem.com)
- DFTB+ (https://www.dftbplus.org)

One needs to use one of these programs, in combination with GENESIS, to perform QM/MM calculations.

For more information on the method and implementation, see Ref. [86].

In order to run QM/MM calculations, users add the **[QMMM]** section in the control file. Avaliable options are listed in the following.

---

qmtyp *DFTB+ / GAUSSIAN / QCHEM / TERACHEM*

The QM program to use in QM/MM calculations.

qmatm_select_index *Integer*

Index of a group of atoms which is treated as QM atoms. Link hydrogen atoms are automatically added based on a bond connectivity (e.g., given by a PSF file). The index must be defined in **[SELECTION]** (see *Selection section*).

qmcnt *Character*

A template input file for QM calculations.

qmexe *Character*

A script file to execute the QM program.

Note: Sample files for **qmcnt** and **qmexe** are available in our github (https://github.com/yagikiyoshi/QMMMscripts)

workdir *Character*

Default : qmmm

The name of a directory where QM input/output files are generated. The replica ID is added after this name, e.g., qmmm.0, qmmm.1, etc.

basename *Character*

Default : N/A

The basename of input / output files of QM calculations.

qmsave_period *Integer*

Default : 1

Frequency to save input / output files for QM calculations.

savedir *Character*

Default : N/A

If present, QM files are copied from **workdir** to this directory. It is typically the case that QM calculations are carried out within a node, and the whole simulation (such as REMD) accross nodes. Then, it is useful for a better performance to set **workdir** to a local disk of each node with fast access (e.g., /dev/shm), and copy the QM files to **savedir** with a frequency specified by qmsave_period.

qmmaxtrial *Integer*

Default : 1

The maximum number of trial run for QM calculations. When a QM calculation fails, **GENESIS** repeats the calculation until the iteration reaches this number. The SCF threshold is lowered, if the SCF threshold option is present in the QM control file.

exclude_charge *ATOM / GROUP*

Default: GROUP

This option specifies how to exclude the MM charge in the vicinity of a QM-MM boundary to avoid overpolarization of QM electron density. *ATOM* excludes only the charge of MM link atom, while *GROUP* excludes the charges of all MM atoms that belongs to the same group as a MM atom at the boundary.

Note: Although the QM/MM function in **GENESIS** is rapidly growing, the current implementation still has some limitations:

- MM must be CHARMM.
- PBC is NOT supported.

Extensions to lift these limitations are on-going. In particular, we will support QM/MM-MD, AMBER force fields, and other QM programs in the near future.

Note: A non-PBC system for QM/MM calculations can be created from MD trajectory (pdb, dcd) using **qmmm_generator** in the analysis tool. See the tutorial of QM/MM (https://www.r-ccs.riken.jp/labs/cbrt/tutorials2019/tutorial-16-1) for more details.

18.2 Examples

In the following example, the atoms from # 1 to 14 are selected as QM atoms by **[SELECTION]** section. The QM program is Gaussian. A directory `qmmm_min` is created, where input and output files for Gaussian (jobXXXX.inp and jobXXXX.log) are saved every 10 steps.

```
[SELECTION]
group1   = atomno:1-14

[QMMM]
qmatm_select_index  = 1
qmtyp               = gaussian
qmcnt               = gaussian.com
qmexe               = runGau.sh
workdir             = qmmm_min
```

(continues on next page)

(continued from previous page)

```
basename             = job
qmsave_period        = 10
```

The following example is for DFTB+. Because DFTB calculations typically take < 1 sec per one snapshot, I/O to generate the input and read output could be non-negligible. It is recommended to set **workdir** to a fast disk such as `/dev/shm`. The input and output files of DFTB+ will be copied to `qmmm_min` every 100 steps.

```
[QMMM]
qmatm_select_index = 1
qmtyp              = dftb+
qmcnt              = dftb.hsd
qmexe              = runDFTB.sh
workdir            = /dev/shm/qmmm_min
savedir            = qmmm_min
basename           = job
qmsave_period      = 100
```

CHAPTER

NINETEEN

VIBRATION SECTION

19.1 Vibrational analysis

Vibration is available only in **ATDYN**.

In the **[VIBRATION]** section, users can specify keywords for molecular vibrational analysis. Vibrational analysis in **GENESIS** is done for a subsystem, i.e., for a molecule of interest in the system. In the subsystem's space, a mass-weighted Hessian matrix is generated and diagonalized to obtain normal modes ($\mathbf{C}$) and harmonic frequencies (ω),

$$\mathbf{HC} = \omega\mathbf{C}$$

where $\mathbf{H}$ is the mass-weighted Hessian matrix,

$$H_{ij} = \frac{1}{\sqrt{m_i m_j}} \frac{\partial^2 V}{\partial x_i \partial x_j},$$

with the mass of the *i*-th atom, m_i, and the potential energy, *V*. The Hessian matrix is calculated by numerical differentiations of the gradient,

$$\frac{\partial^2 V}{\partial x_i \partial x_j} \simeq \frac{1}{2\delta_i}\left(\frac{\partial V(+\delta_i)}{\partial x_j} - \frac{\partial V(-\delta_i)}{\partial x_j}\right)$$

This step requires 6 *N* number of gradient calculations, where *N* is the number of atoms in the subsystem. The gradient calculations are parallelized by distributing over MPI processes.

The information is output to a minfo file, which can be visualized by a molecular vibrational program, SINDO.

---

runmode *HARM / QFF / GRID*

Default : HARM

Specifies the type of calculation. **HARM** invokes the harmonic analysis. **QFF** and **GRID** are options for generating anharmonic potential.

nreplica *Integer*

Default : 1

The number of MPI processes.

vibatm_select_index *Integer*

Default: N/A

Indices of a group of atoms to specify a target subsystem for vibrational analyses. The indices must be defined in **[SELECTION]** (see *Selection section*).

output_minfo_atm *Integer*

Default: N/A

Indices of a group of atoms, which are printed to a minfo file in addition to the target subsystem. This option is useful when one would like to visualize the atoms surrounding the target subsystem. The indices must be defined in **[SELECTION]** (see *Selection section*).

diff_stepsize *Real*

Default : 0.01 (unit: Å)

The size of numerical differentiations when generating the Hessian matrix.

minfo_folder *Character*

Default : minfo.files

The name of a directory where intermediate minfo files are stored. If the directory and intermediate files are present, the program restarts from where it ended in the last run.

Note: The geometry of the subsystem needs to be optimized prior to the vibrational analysis. RMSG < 0.35 kcal/mol/Åis recommended.

Furthermore, anharmonic vibrational calculations can be carried out by combining SINDO and GENESIS. The following two options are used for generating anharmonic potential energy surfaces. For more details, visit the website of SINDO (https://tms.riken.jp/en/research/software/sindo).

---

gridfile *Character*

Default : makeQFF.xyz (for QFF) and **makeGrid.xyz** for GRID

The name of a file containing the XYZ coordinates of grid points for generating the anharmonic PES. The xyz file is generated by MakePES module of SINDO.

datafile *Character*

Default : makeGrid.dat

The name of a file containing the energy, dipole moment, etc. at grid points specified by **gridfile**. The file is used by SINDO for generating GRID potentials.

19.2 Examples

In the following example, the vibrational analysis is performed for a subsystem composed of atom number 5-8 (group1) and residue number 12-14 of segment “WAT” (group2). 4 MPI processes are used to calculate the gradients at grid points of numerical differentiations. The output is written to a minfo file, where the coordinates are given not only for target atoms (group1 and group2) but also for the whole protein (segid PROA).

```
[OUTPUT]
minfofile = qmmm_vib.minfo

[VIBRATION]
runmode             = HARM
nreplica            = 4
vibatm_select_index = 1 2
output_minfo_atm    = 3

[SELECTION]
group1  = atomno:5-8
group2  = segid:WAT and (resno:12-14)
group3  = segid:PROA
```

CHAPTER

TWENTY

EXPERIMENTS SECTION

20.1 Cryo-EM flexible fitting

Cryo-electron microscopy (Cryo-EM) is one of the powerful tools to determine three-dimensional structures of biomolecules at near atomic resolution. Flexible fitting has been widely utilized to model the atomic structure from the experimental density map [87]. One of the commonly used methods is the MD-based flexible fitting [88] [89]. In the method, the total potential energy is defined as the summation of a force field V_{FF} and biasing potential V_{EM} that guides the protein structure towards the target density:

$$V_{\mathrm{total}} = V_{\mathrm{FF}} + V_{\mathrm{EM}}$$

In the *c.c.*-based approach [87], one of the commonly used formulas for V_{EM} is

$$V_{\mathrm{EM}} = k(1 - c.c.)$$

where *k* is the force constant, and *c.c.* is the cross-correlation coefficient between the experimental and simulated EM density maps, calculated as

$$c.c. = \frac{\sum\limits_{ijk} \rho^{\mathrm{exp}}(i,j,k)\rho^{\mathrm{sim}}(i,j,k)}{\sqrt{\sum\limits_{ijk} \rho^{\mathrm{exp}}(i,j,k)^2 \sum\limits_{ijk} \rho^{\mathrm{sim}}(i,j,k)^2}}$$

(*i*, *j*, *k*) is a voxel index in the density map, and ρ^{exp} and ρ^{sim} are the experimental and simulated EM densities, respectively.

The simulated densities are usually computed using a Gaussian mixture model, where a 3D Gaussian function is put on the Cartesian coordinates of each target atom (i.e., protein atom), and all contributions are integrated in each voxel of the map. Here, several schemes have been proposed, in which the Gaussian function is weighted with an atomic number [90] or mass [91], or it is simply applied to non-hydrogen atom [87]. In **GENESIS**, the last scheme is introduced. The simulated density of each voxel is defined as:

$$\rho^{\mathrm{sim}}(i.j,k) = \sum_{n=1}^{N} \int\int\int_{V_{ijk}} g_n(x,y,z)\mathrm{d}x\mathrm{d}y\mathrm{d}z$$

where V_{ijk} is the volume of the voxel, *N* is the total number of non-hydrogen atoms in the system, and *n* is the index of the atom. The Gaussian function $g_n(x,y,z)$ is given by

$$g_n(x, y, z) = \exp\left[-\frac{3}{2\sigma^2}\left\{(x - x_n)^2 + (y - y_n)^2 + (z - z_n)^2\right\}\right]$$

where (x_n, y_n, z_n) are the coordinates of the *n*-th atom. σ determines the width of the Gaussian function, and the generated EM density has the resolution of 2σ in the map.

In **GENESIS**, EM biasing force is treated as a kind of "Restraints" (see *Restraints section*). Therefore, flexible fitting can be combined with various methods such as the replica-exchange umbrella-sampling scheme (REUSfit) [92], all-atom Gō model (MDfit) [93], and GB/SA implicit solvent model. In addition, the method is parallelized with the hybrid MPI+OpenMP scheme in both **ATDYN** and **SPDYN**, and also accelerated with GPGPU calculation in **SPDYN** [94]. In the flexible fitting simulation, the users add the **[EXPERIMENTS]** section in the control file, and specify the following keywords. Note that the **[FITTING]** section (see *Fitting section*) is not related to this cryo-EM flexible fitting.

---

emfit *YES / NO*

Default : NO

Turn on or off the cryo-EM flexible fitting.

emfit_target *Character*

Default : N/A

The file name of the target EM density map. The available file format is MRC/CCP4 (ver. 2000 or later) or SITUS (https://situs.biomachina.org/), which is automatically selected according to the file extension. The file extension should be ".map", ".mrc", or ".ccp4" for MRC/CCP4, and ".sit" for SITUS.

emfit_sigma *Real*

Default : 2.5 (unit : Å)

Resolution parameter of the simulated map. This is usually set to the half of the resolution of the target map. For example, if the target map resolution is 5 Å, "emfit_sigma=2.5" is a reasonable choice.

emfit_tolerance *Real*

Default : 0.001

This variable determines the tail length of the Gaussian function. For example, if "emfit_tolerance=0.001" is specified, the Gaussian function is truncated to zero when it is less than 0.1% of the maximum value. Smaller value requires large computational cost.

emfit_zero_threshold *Real*

Default : 0.0

This variable determines a threshold to set zero-densities in the target EM map. If the density in a voxel of the target map is under a given "emfit_zero_threshold", the density is set to zero.

emfit_period *Integer*

Default : 1

Update frequency of the EM biasing force. In the case of "emfit_period=1", the force is updated every step (slow but accurate).

Note: The force constant of the biasing potential is given in the **[RESTRAINTS]** section in a similar manner as the other restraint potentials, where "functionN = EM" is specified for the restraint type (see *Restraints section*). The unit of the force constant is kcal/mol.

Note: The flexible fitting is available with the all-atom explicit solvent system in a periodic boundary condition. In this case, both protein atoms and target densities must be inside the unit cell of the simulation box. The edge coordinates of the unit cell are always $(X, Y, Z) = 0.5 \times (\pm\text{box\_size\_x}, \pm \text{box\_size\_y}, \pm \text{box\_size\_z})$, namely, center of the unit cell is the origin (0,0,0).

20.2 Examples

The following is an example of the cryo-EM flexible fitting using *k* = 10,000 kcal/mol for the 4.1 Åresolution map. The other sections are common to the conventional MD simulations.

```
[SELECTION]
group1          = all and not hydrogen

[RESTRAINTS]
nfunctions      = 1
function1       = EM                # apply EM biasing potential
constant1       = 10000             # force constant in Eem = k*(1 - c.c.)
select_index1   = 1                 # apply force on protein heavy atoms

[EXPERIMENTS]
emfit           = YES               # perform EM flexible fitting
emfit_target    = emd_8623.sit      # target EM density map
emfit_sigma     = 2.05              # half of the map resolution (4.1 A)
emfit_tolerance = 0.001             # Tolerance for error (0.1%)
emfit_period    = 1                 # emfit force update period
```

The following is an example of REUSfit using 8 replicas, where the force constants 100–800 kcal/mol are assigned to each replica, and exchanged during the simulation (see also *REMD section*).

```
[REMD]
dimension       = 1
exchange_period = 1000
type1           = RESTRAINT
nreplica1       = 8
rest_function1  = 1

[SELECTION]
group1          = all and not hydrogen

[RESTRAINTS]
```

(continues on next page)

(continued from previous page)

```
nfunctions      = 1
function1       = EM
constant1       = 100 200 300 400 500 600 700 800
select_index1   = 1

[EXPERIMENTS]
emfit           = YES
emfit_target    = target.sit
emfit_sigma     = 5
emfit_tolerance = 0.001
emfit_period    = 1
```

CHAPTER TWENTYONE

ALCHEMY SECTION

21.1 Free energy perturbation

In the **[Alchemy]** section, the users can specify keywords for the free-energy perturbation (FEP) method, which is one of the alchemical free energy calculations. The FEP method calculates the free-energy difference between two states by gradually changing a part of the system from one state to another state. Since the free energy depends only on the initial and final states, any intermediate states can be chosen, regardless of whether they are physically realizable or not. If the intermediate states are not physically unrealizable but computationally realizable, the calculation and thermodynamic process are referred to as the alchemical free-energy calculation and alchemical process, respectively. Using alchemical processes, the FEP method can be applied to a variety of free-energy calculations, such as solvation free energies, binding free energies, and free-energy changes upon protein mutations. In particular, absolute binding free energies can be calculated by gradually vanishing a ligand of interest, while relative binding free energies can be calculated by gradually changing one ligand to another.

GENESIS enables to perform various alchemical calculations by implementing dual-topology and hybrid-topology schemes, soft-core potentials, and lambda-exchange calculations based on REMD. GPGPU acceleration are also available in FEP simulations. The short-range non-bonded interactions (LJ and PME real part) are calculated by GPU, while the long-range (PME reciprocal part) are calculated by CPU. Currently, **SPDYN** supports only CHARMM and AMBER force fields for FEP simulations. The FEP method is not available in **ATDYN**. In this section, the FEP functions in GENESIS are briefly summarized and some examples are shown.

21.1.1 Theory of FEP

The FEP method calculates the free-energy difference between two states, A and B, using the following equation.

$$\begin{aligned}
\Delta F &= F_{\mathrm{B}} - F_{\mathrm{A}} \\
&= -k_{\mathrm{B}T} \ln \frac{\int dx \exp[-\beta U_{\mathrm{B}}(x)]}{\int dx \exp[-\beta U_{\mathrm{A}}(x)]} \\
&= -k_{\mathrm{B}T} \ln \frac{\int dx \exp[-\beta U_{\mathrm{A}}(x) - \beta \Delta U(x)]}{\int dx \exp[-\beta U_{\mathrm{A}}(x)]} \\
&= -k_{\mathrm{B}T} \ln \langle \exp[-\beta \Delta U(x)] \rangle_{\mathrm{A}} ,
\end{aligned}$$

where F and U are the free energy and potential energy of state A or B, ΔU is the difference between U_{A} and U_{B}, and x is the configuration of the system. The bracket at the final line represents the ensemble average at state A. This equation means that ΔF can be estimated by sampling only equilibrium configurations of the state A. However, if the difference between states A and B is large, there is little overlap of energy distributions (Left of Fig. 21.1) and the configurations at state B are poorly sampled by simulations at the state A, leading to large statistical errors. To reduce the errors, $n-2$ intermediate states are inserted between states A and B to overlap energy distributions (Right of Left of Fig. 21.1). The potential energy of the intermediate state i is defined by $U(\lambda_i) = (1-\lambda_i)U_\mathrm{A} + \lambda_i U_\mathrm{B}$, where λ_i is the scaling parameter for connecting the initial and final states. By changing λ_i from A to B, states A and B can be connected smoothly. ΔF can be estimated by calculating the summation of free-energy changes between adjacent states:

$$\begin{aligned}\Delta F &= \sum_{i=0}^{n-1} \Delta F_i \\ &= -k_B T \sum_{i=0}^{n-1} \ln \left\langle \exp[-\beta(U(\lambda_i+1) - U(\lambda_i))] \right\rangle ,\end{aligned}$$

where the subscript λ_i represents the ensemble average at state i. The states of $i=0$ and n correspond to state A and B, respectively.

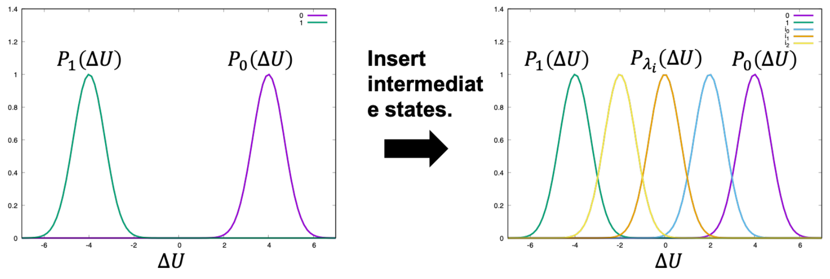


Fig. 21.1: Insertion of intermediate states. Some intermediate states are inserted between the reference state and the target state states to overlap energy distributions.

21.1.2 Dual topology scheme

One of the most important applications of the FEP method is the calculation of protein-ligand binding affinity, which represents how strong a ligand binds to a protein. In drug discovery, it is required to find a ligand that best binds to the target protein from a large number of chemical compounds. The difference between binding affinities of two ligands, called the relative binding affinity, can be calculated by changing one ligand into another ligand during the simulation. For example, consider the mutation from benzene to phenol (Fig. 21.2 (a)). Benzene and phenol correspond to states A and B, respectively. The atoms of benzene except for a hydrogen atom are common to both ligands, which have no need to be perturbed. On the other hand, the H atom of benzene and the OH atoms of phenol are different in not only their force field parameters but also their topology. To minimize perturbation and treat the topological difference, topologies of two ligands are unified such that the atoms with different topologies connect with the common atoms (Fig. 21.2 (b)). This topology is called the dual topology, which consists of the common atoms, the atoms included in only state A (dualA in Fig. 21.2 (b)), and the atoms included in only state B (dualB in Fig. 21.2 (b)) [95][96][97]. The perturbation is applied to only the dualA and dualB parts.

The free-energy change upon the mutation can be calculated by gradually switching the interactions of the dual-topology part from benzene to phenol (Fig. 21.2 (c)). At state A, only the H atom exists in the dual-topology part, while the OH atoms do not interact with the other atoms in the system. During the alchemical transformation, the H atom gradually disappears, whereas the OH atoms gradually appears. At state B, only the OH atoms exist in the dual-topology part and interact with the other atoms. The

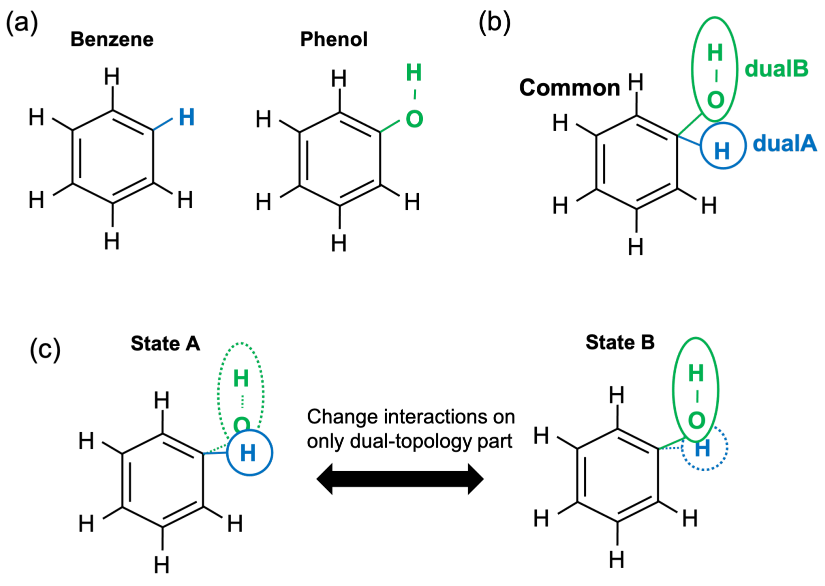


Fig. 21.2: Dual topology scheme. (a) Benzene and phenol. Common atoms, H atom of benzene, and OH atoms of phenol are in black, cyan, and green, respectively. (b) Dual topology of benzene and phenol. (c) The alchemical transformation from benzene to phenol.

non-bonded potential energy is modified to connect smoothly state A to state B by introducing λ_{LJ} and λ_{elec}:

$$\begin{aligned}U_{\text{nonbond}} = {} & U_{\text{LJ}}^{\text{other-other}} + U_{\text{LJ}}^{\text{other-common}} + U_{\text{LJ}}^{\text{common-common}} + U_{\text{elec}}^{\text{other-other}} + U_{\text{elec}}^{\text{other-common}} + U_{\text{elec}}^{\text{common-common}} \\ & + \lambda_{\text{LJ}}^{A}(U_{\text{LJ}}^{\text{other-dualA}} + U_{\text{LJ}}^{\text{common-dualA}} + U_{\text{LJ}}^{\text{dualA-dualA}}) \\ & + \lambda_{\text{LJ}}^{B}(U_{\text{LJ}}^{\text{other-dualB}} + U_{\text{LJ}}^{\text{common-dualB}} + U_{\text{LJ}}^{\text{dualB-dualB}}) \\ & + \lambda_{\text{elec}}^{A}(U_{\text{elec}}^{\text{other-dualA}} + U_{\text{elec}}^{\text{common-dualA}} + U_{\text{elec}}^{\text{dualA-dualA}}) \\ & + \lambda_{\text{elec}}^{B}(U_{\text{elec}}^{\text{other-dualB}} + U_{\text{elec}}^{\text{common-dualB}} + U_{\text{elec}}^{\text{dualB-dualB}})\end{aligned}$$

where “common”, “dualA”, “dualB”, and “other” in the superscripts respectively represent the common atoms, the atoms existing only at state A, the atoms existing only at state B, and the other molecules including solvent molecules, proteins, or other ligands. For example, $U_{\text{LJ}}^{\text{common-dualA}}$ represents the LJ interaction between a common atom and a dualA atom. The potential energy at $\lambda_{\text{LJ}}^{A} = 1$, $\lambda_{\text{elec}}^{A} = 1$, $\lambda_{\text{LJ}}^{B} = 0$, and $\lambda_{\text{elec}}^{B} = 0$ corresponds to that of state A, while the energy at $\lambda_{\text{LJ}}^{A} = 0$, $\lambda_{\text{elec}}^{A} = 0$, $\lambda_{\text{LJ}}^{B} = 1$, and $\lambda_{\text{elec}}^{B} = 1$ corresponds to that of state B. By gradually changing λ_{LJ}^{A}, $\lambda_{\text{elec}}^{A}$, λ_{LJ}^{B}, and $\lambda_{\text{elec}}^{B}$, states A and B can be connected smoothly.

In GENESIS, the dual-topology scheme is available by specifying **fep_topology = Dual** in the [ALCHEMY] section. The users should select the atoms of the dual-topology part in the [SELECTION] section and assign their group numbers to **dualA** and **dualB** in the [ALCHEMY] section. λ_{LJ}^{A}, $\lambda_{\text{elec}}^{A}$, λ_{LJ}^{B}, and $\lambda_{\text{elec}}^{B}$ can be specified by **lambljA**, **lambljB**, **lambelA**, and **lambelB**, respectively. An example is shown below.

```
[ALCHEMY]
fep_topology = Dual
dualA        = 1   # group1 in [SELECTION]
dualB        = 2   # group2 in [SELECTION]
lambljA      = 1.00 0.75 0.50 0.25 0.00
lambljB      = 0.00 0.25 0.50 0.75 1.00
```

(continues on next page)

(continued from previous page)

```
lambelA       = 1.00 0.75 0.50 0.25 0.00
lambelB       = 0.00 0.25 0.50 0.75 1.00

[SELECTION]
group1        = ai:1    # atoms in dual A
group2        = ai:2-3  # atoms in dual B
```

In this example, five sets of λ^A_{LJ}, $\lambda^A_{\mathrm{elec}}$, λ^B_{LJ}, and $\lambda^B_{\mathrm{elec}}$ are used to connect state A to state B. In the [SELECTION] section, the H atom of benzene and the OH atoms of phenol are selected by group 1 and 2, respectively. The group IDs are specified as **dualA = 1** and **dualB = 2**.

21.1.3 Hybrid topology scheme

In the dual-topology scheme, the force field parameters of common atoms are assumed to be the same in both states. However, in general, they are different each other. Fig. 21.3 (a) shows that the charge distribution of the benzene ring of benzene is different from that of phenol. The parameters of the common atoms for bond, angle, and dihedral are also different between two molecules. To treat the difference of the force field parameters, the parts of the molecules with the same topology are superimposed (Fig. 21.3 (b)) [98]. The superimposed part has a single topology, in which the parts corresponding to states A and B are referred to as "singleA" and "singleB", respectively. During FEP simulations, the single-topology part does not change its topology, but its force field parameters (charge, LJ, and internal bond) are gradually changed from state A to state B (Fig. 21.3 (c)). In contrast, the other part has a dual topology, in which "dualA" and "dualB" correspond to states A and B, respectively. In the dual-topology part, their topology is changed as well as their parameters (Fig. 21.3 (c)).

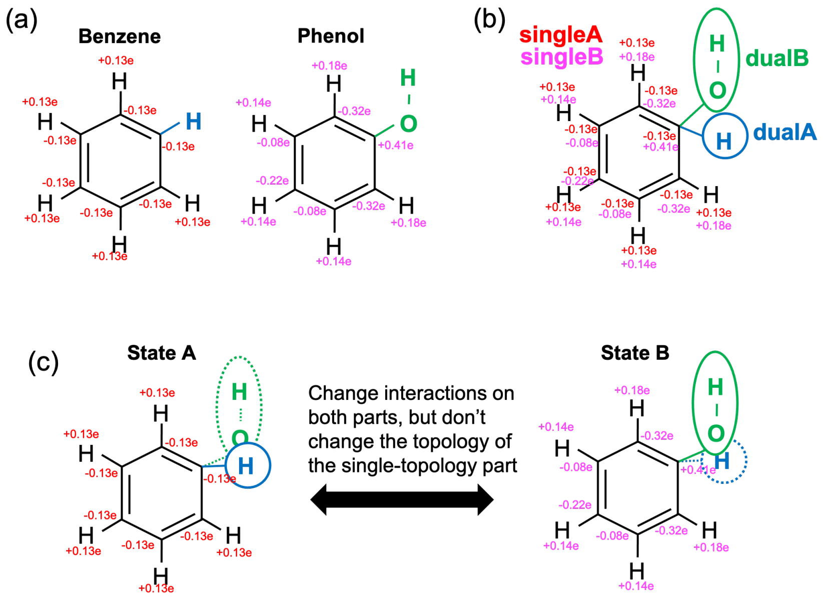

(a)
Benzene
Phenol
(b)
singleA
singleB
dualB
dualA
(c)
State A
Change interactions on both parts, but don't change the topology of the single-topology part
State B

Fig. 21.3: Hybrid topology scheme. (a) Benzene and phenol. H atom of benzene and OH atoms of phenol are in cyan and green, respectively. The point charges on common atoms are shown in red and magenda, which are determined using Amber Tools [1]. (b) Hybrid topology of benzene and phenol. (c) The alchemical transformation from benzene to phenol.

In the hybrid topology scheme, the potential energy is scaled by λ_{LJ}, λ_{elec}, and λ_{bond}:

$$
\begin{aligned}
U_{\text{nonbond}} = {} & U_{\text{LJ}}^{\text{other-other}} + U_{\text{elec}}^{\text{other-other}} \\
& + \lambda_{\text{LJ}}^{A}(U_{\text{LJ}}^{\text{other-singleA}} + U_{\text{LJ}}^{\text{other-dualA}} + U_{\text{LJ}}^{\text{singleA-singleA}} + U_{\text{LJ}}^{\text{singleA-dualA}} + U_{\text{LJ}}^{\text{dualA-dualA}}) \\
& + \lambda_{\text{LJ}}^{B}(U_{\text{LJ}}^{\text{other-singleB}} + U_{\text{LJ}}^{\text{other-dualB}} + U_{\text{LJ}}^{\text{singleB-singleB}} + U_{\text{LJ}}^{\text{singleB-dualB}} + U_{\text{LJ}}^{\text{dualB-dualB}}) \\
& + \lambda_{\text{elec}}^{A}(U_{\text{elec}}^{\text{other-singleA}} + U_{\text{elec}}^{\text{other-dualA}} + U_{\text{elec}}^{\text{singleA-singleA}} + U_{\text{elec}}^{\text{singleA-dualA}} + U_{\text{elec}}^{\text{dualA-dualA}}) \\
& + \lambda_{\text{elec}}^{B}(U_{\text{elec}}^{\text{other-singleB}} + U_{\text{elec}}^{\text{other-dualB}} + U_{\text{elec}}^{\text{singleB-singleB}} + U_{\text{elec}}^{\text{singleB-dualB}} + U_{\text{elec}}^{\text{dualB-dualB}})
\end{aligned}
$$

$$
\begin{aligned}
U_{\text{bond}} = {} & U_{\text{bond}}^{\text{other}} + U_{\text{bond}}^{\text{dualA}} + U_{\text{bond}}^{\text{dualB}} + U_{\text{bond}}^{\text{singleA-dualA}} + U_{\text{bond}}^{\text{singleB-dualB}} \\
& + \lambda_{\text{bond}}^{A}(U_{\text{bond}}^{\text{singleA}}) \\
& + \lambda_{\text{bond}}^{B}(U_{\text{bond}}^{\text{singleB}})
\end{aligned}
$$

where "singleA", "singleB", "dualA", and "dualB" in the superscripts respectively represent the atoms corresponding to "singleA", "singleB", "dualA", and "dualB" parts, respectively, and "other" represents the other molecules including solvent molecules, proteins, or other ligands. The potential energy at $\lambda_{\text{LJ}}^{A} = 1$, $\lambda_{\text{elec}}^{A} = 1$, $\lambda_{\text{bond}}^{A} = 1$, $\lambda_{\text{LJ}}^{B} = 0$, $\lambda_{\text{elec}}^{B} = 0$, and $\lambda_{\text{bond}}^{B} = 0$ corresponds to that of state A, while the energy at $\lambda_{\text{LJ}}^{A} = 0$, $\lambda_{\text{elec}}^{A} = 0$, $\lambda_{\text{bond}}^{A} = 0$, $\lambda_{\text{LJ}}^{B} = 1$, $\lambda_{\text{elec}}^{B} = 1$, and $\lambda_{\text{bond}}^{B} = 1$ corresponds to that of state B. By gradually changing the lambda values, states A and B can be connected smoothly.

In GENESIS, the hybrid-topology scheme is available by specifying **fep_topology = Hybrid** in the [ALCHEMY] section. The users should select the atoms of the single-topology and dual-topology parts in the [SELECTION] section and assign their group numbers to **singleA**, **singleB**, **dualA**, and **dualB** in the [ALCHEMY] section. λ_{LJ}^{A}, $\lambda_{\text{elec}}^{A}$, $\lambda_{\text{bond}}^{A}$, λ_{LJ}^{B}, $\lambda_{\text{elec}}^{B}$, and $\lambda_{\text{bond}}^{B}$ can be specified by **lambljA**, **lambljB**, **lambelA**, and **lambelB**, respectively. An example is shown below.

```
[ALCHEMY]
fep_topology = Hybrid
singleA      = 1  # group1 in [SELECTION]
singleB      = 2  # group2 in [SELECTION]
dualA        = 3  # group3 in [SELECTION]
dualB        = 4  # group4 in [SELECTION]
lambljA      = 1.00 0.75 0.50 0.25 0.00
lambljB      = 0.00 0.25 0.50 0.75 1.00
lambelA      = 1.00 0.75 0.50 0.25 0.00
lambelB      = 0.00 0.25 0.50 0.75 1.00
lambbondA    = 1.00 0.75 0.50 0.25 0.00
lambbondB    = 0.00 0.25 0.50 0.75 1.00

[SELECTION]
group1       = ai:1-11   # atoms in single A
group2       = ai:13-23  # atoms in single B
group3       = ai:12     # atoms in dual A
group4       = ai:24-25  # atoms in dual B
```

In this example, five sets of the lambda values are used to connect state A to state B. In the [SELECTION] section, the benzene ring of benzene, the benzene ring of phenol, the H atom of benzene, and the OH atoms of phenol are selected by group 1, 2, 3, and 4, respectively. The group IDs are specified as **singleA = 1**, **singleB = 2**, **dualA = 3**, and **dualB = 4**.

21.1.4 Soft core potentials

Close to the end point of alchemical calculations ($\lambda_{\mathrm{LJ}} = 0$ or 1), overlaps between perturbed atoms or between perturbed and non-perturbed atoms occur, causing large energy change. The system becomes unstable due to the overlaps and the simulations might be stopped, which is called the end point catastrophe. To avoid the catastrophe, the soft core is introduced to the LJ potential [99]:

$$U_{\mathrm{LJ}}(r_{ij}, \lambda_{\mathrm{LJ}}) = 4\lambda_{\mathrm{LJ}}\epsilon\left[\left(\frac{\sigma^2}{r_{ij}^2 + \alpha_{sc}(1-\lambda_{\mathrm{LJ}})}\right)^6 - \left(\frac{\sigma^2}{r_{ij}^2 + \alpha_{sc}(1-\lambda_{\mathrm{LJ}})}\right)^3\right],$$

where α_{sc} is the parameter for the soft-core potential. In the potential, r_{ij}^2 is shifted to $\alpha_{\mathrm{sc}}(1-\lambda_{\mathrm{LJ}})$, which weakens the repulsive part in the LJ potential when λ_{LJ} approaches 0 (Fig. 21.4). Since the soft-core potential corresponds to the original LJ potential at the end point of λ_{LJ}:

$$U_{\mathrm{LJ}}(r_{ij}, \lambda_{\mathrm{LJ}} = 1) = 4\epsilon\left[\left(\frac{\sigma}{r_{ij}}\right)^{12} - \left(\frac{\sigma}{r_{ij})}\right)^6\right],$$
$$U_{\mathrm{LJ}}(r_{ij}, \lambda_{\mathrm{LJ}} = 0) = 0,$$

the soft-core modification in the LJ potential does not affect the free-energy calculation. α_{sc} can be specified by a keyword **sc_alpha** in the GENESIS control file.

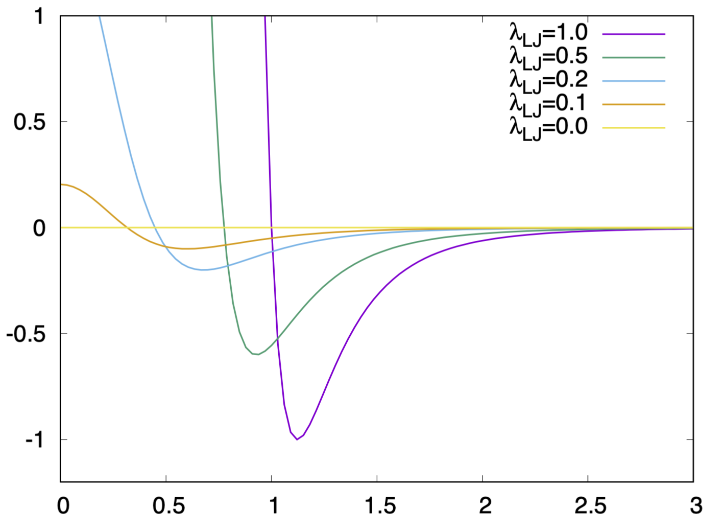


Fig. 21.4: Soft core potential for Lennard-Jones interaction.

In GENESIS, the soft core is also applied to the electrostatic potential [100]:

$$U_{\mathrm{elec}}(r_{ij}, \lambda_{\mathrm{elec}}) = \lambda_{\mathrm{elec}}\frac{q_i q_j \mathrm{erfc}(\alpha\sqrt{r_{ij}^2 + \beta_{sc}(1-\lambda_{\mathrm{elec}})})}{\epsilon\sqrt{r_{ij}^2 + \beta_{sc}(1-\lambda_{\mathrm{elec}}}} + \lambda_{\mathrm{elec}}(\text{PME reciprocal and self terms}),$$

where β_{sc} is the parameter for the electrostatic soft-core potential. In the potential, r_{ij}^2 is also shifted to $\beta_{\mathrm{sc}}(1-\lambda_{\mathrm{elec}})$ like the LJ soft-core potential, which softens disruptions due to overlaps of point charges. This soft-core potential is almost the same as used in Amber [100]. β_{sc} can be specified by a keyword **sc_beta** in the GENESIS control file.

21.1.5 FEP/λ-REMD

To enhance the sampling efficiency, the FEP simulations at different λ values are coupled using the Hamiltonian replica exchange method [64][65]. This method is called FEP/λ-REMD or λ-exchange FEP [69]. Replicas run in parallel and exchange their parameters at fixed intervals during the simulation. The exchanges between adjacent replicas are accepted or rejected according to Metropolis's criterion. In FEP/λ-REMD simulations, [REMD] section is also required. **type1 = alchemy** is set to exchange the lambda values. The following is an example of the control file for the FEP/λ-REMD simulation.

```
[REMD]
dimension       = 1
exchange_period = 500
type1           = alchemy
nreplica1       = 5

[ALCHEMY]
fep_topology = Hybrid
singleA      = 1   # group1 in [SELECTION]
singleB      = 2   # group2 in [SELECTION]
dualA        = 3   # group3 in [SELECTION]
dualB        = 4   # group4 in [SELECTION]
lambljA      = 1.00 0.75 0.50 0.25 0.00
lambljB      = 0.00 0.25 0.50 0.75 1.00
lambelA      = 1.00 0.75 0.50 0.25 0.00
lambelB      = 0.00 0.25 0.50 0.75 1.00
lambbondA    = 1.00 0.75 0.50 0.25 0.00
lambbondB    = 0.00 0.25 0.50 0.75 1.00

[SELECTION]
group1       = ai:1-11    # atoms in single A
group2       = ai:13-23   # atoms in single B
group3       = ai:12      # atoms in dual A
group4       = ai:24-25   # atoms in dual B
```

21.2 Parameters for alchemy section

fep_direction *Bothsides / Forward / Reverse*

Default: Bothsides

Direction of calculation of energy difference. When the current state of the simulation or the replica is i, GENESIS outputs the energy difference between state i and the adjacent state at a frequency determined by **fepout_period**. If **fep_direction = Forward**, the energy difference between states i and $i+1$ is output. If **fep_direction = Reverse**, the energy difference between states i and $i-1$ is output. If **fep_direction = Bothsides**, the energy differences between states i and $i-1$ and between states i and $i+1$ are output.

fepout_period *Integer*

Default: 0

Period of outputting energy differences.

fep_topology *Dual / Hybrid*

Default: Hybrid

Topology of perturbed region. If **fep_topology = Dual**, the dual-topology scheme is used. If **fep_topology = Hybrid**, the hybrid-topology scheme is used.

singleA *Integer* or *None*

Default: 0

Group index for the single topology region of state A. If **singleA = None**, calculations for the region are skipped. If **fep_topology = Hybrid**, this parameter must be specified.

singleB *Integer* or *None*

Default: 0

Group index for the single topology region of state B. If **singleB = None**, calculations for the region are skipped. If **fep_topology = Hybrid**, this parameter must be specified.

dualA *Integer* or *None*

Default: 0

Group index for the dual topology region of state A. If **dualA = None**, calculations for the region are skipped.

dualB *Integer* or *None*

Default: 0

Group index for the dual topology region of state B. If **dualB = None**, calculations for the region are skipped.

sc_alpha *Real*

Default: 5.0 (dimensionless)

Parameter for the soft-core potential for the Lennard-Jones interaction.

sc_beta *Real*

Default: 0.5 (dimensionless)

Parameter for the soft-core potential for the electrostatic interaction.

equilsteps *Integer*

Default: 0

Number of steps of equilibration at each lambda window. If **equilsteps > 0**, equilibration run is performed until the time step reaches **equilsteps**. During the equilibration, energy differences are not outputted. After the equilibration, production run is performed with outputting energy differences until the time step reaches **timesteps + equilsteps**.

lambljA *Real*

Default: 1.0

Scaling parameters for Lennard-Jones interactions in state A ($\lambda_{\mathrm{LJ}}^{A}$).

lambljB *Real*

Default: 1.0

Scaling parameters for Lennard-Jones interactions in state B ($\lambda_{\mathrm{LJ}}^{B}$).

lambelA *Real*

Default: 1.0

Scaling parameters for electrostatic interactions in state A ($\lambda^{A}_{\mathrm{elec}}$).

lambelB *Real*

Default: 1.0

Scaling parameters for electrostatic interactions in state B ($\lambda^{B}_{\mathrm{elec}}$).

lambbondA *Real*

Default: 1.0

Scaling parameters for bonded interactions in state A ($\lambda^{A}_{\mathrm{bond}}$).

lambbondB *Real*

Default: 1.0

Scaling parameters for bonded interactions in state B ($\lambda^{B}_{\mathrm{bond}}$).

lambrest *Real*

Default: 1.0

Scaling parameters for restraint interactions (λ_{rest}).

fep_md_type *Serial / Single / Parallel*

Default: Serial

Type of FEP simulation. If **fep_md_type = Serial**, FEP simulations are performed with changing lambda values specified in **lambljA**, **lambljB**, **lambelA**, etc. For example, if 0.0, 0.5, and 1.0 are specified in **lambljA**, GENESIS first performs the FEP simulation with **lambljA = 0.0**, subsequently performs the FEP simulation with **lambljA = 0.5**, and finally performs the FEP simulation with **lambljA = 1.0**. If **fep_md_type = Single**, a FEP simulation is performed with the lambda window specified in **ref_lambid**. If **fep_md_type = Parallel**, each lambda window is simulated in parallel. In this case, **[REMD]** section must be specified.

ref_lambid *Integer*

Default: 0

Reference window id for a single FEP MD simulation. If **fep_md_type = Single**, **ref_lambid** must be specified.

21.3 Examples

Example of a calculation of the solvation free energy of a ligand. The solvation free energy corresponds to the free-energy change upon the transfer of the ligand from vacuum to solvent. In state A (= in solvent) the ligand fully interacts with solvent molecules, whereas in state B (= in vacuum) those interactions vanishes. To perform such calculation, the dual topology is employed, and **dualA** is set to the group ID of the selected ligand, while **dualB** is set to **NONE**. **dualB = NONE** means that there is no ligand in the system at state B. **lambljA**, **lambljB**, **lambelA**, and **lambelB** should be zero at state B.

```
[ALCHEMY]
fep_direction = BothSides
fep_topology  = Dual
singleA       = NONE
singleB       = NONE
dualA         = 1
dualB         = NONE
fepout_period = 500
equilsteps    = 0
sc_alpha      = 5.0
sc_beta       = 0.5
lambljA       = 1.000 1.000 1.000 1.000 1.000 0.750 0.500 0.250 0.000
lambljB       = 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
lambelA       = 1.000 0.750 0.500 0.250 0.000 0.000 0.000 0.000 0.000
lambelB       = 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
lambbondA     = 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
lambbondB     = 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
lambrest      = 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000

[SELECTION]
group1 = segid:LIG
```

Example of an alchemical transformation between two ligands using serial FEP simulations. When the [REMD] section is not specified and more than one lambda values are specified, GENESIS performs serial calculations by changing lambda values. If **fep_direction** is **Bothsides**, **lambljA**, **lambljB**, **lambelA**, **lambelB**, **lambbondA**, and **lambbondB** are first set to the leftmost values, which are "1.00", "0.00", "1.00", "0.00", "1.00", and "0.00", respectively, in the below example. After the **equilsteps + nsteps**-steps FEP simulation is performed with the set of the lambda values, the lambda values are changed to the second values from the left. In this way, GENESIS performs FEP simulations, changing lambda values. When the FEP simulation with the rightmost values of lambda finishes, GENESIS stops the calculation.

```
[ALCHEMY]
fep_direction = BothSides
fep_topology  = Hybrid
singleA       = 1  # group1 in [SELECTION]
singleB       = 2  # group2 in [SELECTION]
dualA         = 3  # group3 in [SELECTION]
dualB         = 4  # group4 in [SELECTION]
fepout_period = 500
equilsteps    = 0
sc_alpha      = 5.0
sc_beta       = 5.0
lambljA       = 1.00 0.75 0.50 0.25 0.00
lambljB       = 0.00 0.25 0.50 0.75 1.00
lambelA       = 1.00 0.75 0.50 0.25 0.00
lambelB       = 0.00 0.25 0.50 0.75 1.00
lambbondA     = 1.00 0.75 0.50 0.25 0.00
lambbondB     = 0.00 0.25 0.50 0.75 1.00

[SELECTION]
group1        = ai:1-11    # atoms in single A
group2        = ai:13-23   # atoms in single B
group3        = ai:12      # atoms in dual A
group4        = ai:24-25   # atoms in dual B
```

Example of an alchemical transformation between two ligands at a set of lambda values. If the user wants to perform a FEP simulation with a specified set of lambda values, set **fep_md_type** to **Single** and assign the ID of the set of lambda values to **ref_lambid**. In the following example, **ref_lambid** is set to 3, which means that the third column of the lambda values: **lambljA = 0.5**, **lambljB = 0.5**, **lambelA = 0.5**, **lambelB = 0.5**, **lambbondA = 0.5**, and **lambbondB = 0.5**. If **fep_direction = Bothsides**, the energy differences between "**ref_lambid**"-th and "**ref_lambid** -1"-th columns and between "**ref_lambid**"-th and "**ref_lambid** +1"-th columns are outputted into the fepout file. By using these function, the user can independently perform FEP simulations with different lambda values in parallel.

```
[ALCHEMY]
fep_direction   = BothSides
fep_topology    = Hybrid
fep_md_type     = Single
ref_lambid      = 3
singleA         = 1  # group1 in [SELECTION]
singleB         = 2  # group2 in [SELECTION]
dualA           = 3  # group3 in [SELECTION]
dualB           = 4  # group4 in [SELECTION]
fepout_period   = 500
equilsteps      = 0
sc_alpha        = 5.0
sc_beta         = 5.0
lambljA         = 1.00 0.75 0.50 0.25 0.00
lambljB         = 0.00 0.25 0.50 0.75 1.00
lambelA         = 1.00 0.75 0.50 0.25 0.00
lambelB         = 0.00 0.25 0.50 0.75 1.00
lambbondA       = 1.00 0.75 0.50 0.25 0.00
lambbondB       = 0.00 0.25 0.50 0.75 1.00

[SELECTION]
group1          = ai:1-11   # atoms in single A
group2          = ai:13-23  # atoms in single B
group3          = ai:12     # atoms in dual A
group4          = ai:24-25  # atoms in dual B
```

Example of an alchemical transformation between two ligands using a parallel FEP simulation. When the [REMD] section is specified, GENESIS performs the FEP/λ-REMD simulation. Each lambda value in **lambljA**, **lambljB**, **lambelA**, **lambelB**, **lambbondA**, and **lambbondB** is assigned to each replica, and the FEP simulation in each replica is performed in parallel. The lambda values are exchanged at fixed intervals specified by **exchange_period** during the simulation.

```
[REMD]
dimension       = 1
exchange_period = 1000
type1           = alchemy
nreplica1       = 5

[ALCHEMY]
fep_direction   = BothSides
fep_topology    = Hybrid
singleA         = 1  # group1 in [SELECTION]
singleB         = 2  # group2 in [SELECTION]
dualA           = 3  # group3 in [SELECTION]
dualB           = 4  # group4 in [SELECTION]
fepout_period   = 500
```

(continues on next page)

(continued from previous page)

```
equilsteps       = 0
sc_alpha         = 5.0
sc_beta          = 5.0
lambljA          = 1.00 0.75 0.50 0.25 0.00
lambljB          = 0.00 0.25 0.50 0.75 1.00
lambelA          = 1.00 0.75 0.50 0.25 0.00
lambelB          = 0.00 0.25 0.50 0.75 1.00
lambbondA        = 1.00 0.75 0.50 0.25 0.00
lambbondB        = 0.00 0.25 0.50 0.75 1.00

[SELECTION]
group1           = ai:1-11   # atoms in single A
group2           = ai:13-23  # atoms in single B
group3           = ai:12     # atoms in dual A
group4           = ai:24-25  # atoms in dual B
```

CHAPTER

TWENTYTWO

TROUBLE SHOOTING

The followings are representative error messages that the users can frequently encounter during the simulations. We describe possible reasons for each error message, and provide suggestions to solve the problem.

Compute_Shake> SHAKE algorithm failed to converge

This message indicates that constraint for the rigid bond using the SHAKE algorithm (see *Constraints section*) was failed due to some reasons. In most cases, SHAKE errors are originated from insufficient equilibration, bad initial structure, or bad input parameters. We recommend the users to check the following points:

- Reconsider the equilibration scheme. More moderate equilibration might be needed. For example, heating the system from 0 K, using a shorter timestep (e.g., 1.0 fs), or performing long energy minimization is a possible solution.
- Check the initial structure very carefully. One of the frequent mistakes in the initial structure modeling is "ring penetration" of covalent bonds. One covalent bond might be somehow inserted into an aromatic ring. Solve the ring penetration first, and then try the simulation again.
- Some force field parameters are missing or wrong, which can easily cause unstable simulations.

Check_Atom_Coord> Some atoms have large clashes

This message indicates that there is an atom pair whose distance is zero or close to zero. Those atom indexes and distance are displayed in a warning message: "WARNING: too short distance:". This situation is not allowed, especially in **SPDYN**, since it can cause a numerical error in the lookup table method. Check the initial structure first. Even if you cannot see such atomic clashes, there may be a clash between the atoms in the unit cell and image cells in the case of the periodic boundary condition. One of the automatic solutions is to specify "contact_check = YES" in the control file (see *Energy section*). However, this cannot work well, if the distance is exactly zero. In such cases, the problem should be solved by the users themselves. For example, the users may have to slightly move the clashing atoms manually, or specify larger or smaller box size, or rebuild the initial structure more carefully.

Setup_Processor_Number> Cannot define domains and cells. Smaller MPI processors, or shorter pairlistdist, or larger boxsize should be used

This message indicates that the total number of MPI processors used in your calculation is not appropriate for your system. The users had better understand relations between the system size and number of MPI processors. In **SPDYN**, the system is divided into several domains for parallel computation, where the number of domains must be equal to the number of MPI processors (see *Available Programs*). In most cases, this message tells you that the system could not be divided into the specified number of domains. Although there are mainly three solutions for this problem, first one is the most recommended way:

- Use smaller number of MPI processors. If it can work, the previous number was too large to handle the system.
- Use shorter pairlistdist. This treatment can make a domain size smaller, allowing to use a larger number of MPI processors. However, this is not recommended, if you are already using a recommended parameter set for switchdist, cutoffdist, and pairlistdist (e.g., 10, 12, and 13.5 Å in the CHARMM force field)
- Build a larger initial structure by adding solvent molecules in the system, which may allow the users to divide the system into the desired number of domains.

Update_Boundary_Pbc> too small boxsize/pairdist. larger boxsize or shorter pairdist should be used.

This message indicates that your system is too small to handle in the periodic boundary condition. In **ATDYN**, cell-linked list method is used to make non-bonded pairlists, where the cell size is determined to be close to and larger than the pairlist distance given in the control file. In addition, the total number of cells in x, y, and z dimensions must be at least three. **SPDYN** has a similar lower limitation in the available box size. Therefore, in order to solve this problem, the users may have to set a shorter pairlistdist, or build a larger system by adding much solvent molecules.

Compute_Energy_Experimental_Restraint_Emfit> Gaussian kernel is extending outside the map box

This message indicates that the simulated densities were generated outside the target density map. If atoms to be fitted are located near the edge of the target density map, this error can frequently happen.

- Create a larger density map by adding an enough margin to the map, which can be easily accomplished with the "voledit" tool in SITUS (https://situs.biomachina.org/).
- Examine a normal MD simulation by turning off the EM biasing potential (emfit = NO). If the simulation is not stable, there is an issue in the molecular mechanics calculation rather than the biasing potential calculation. In such cases, please check the initial structure carefully. There might be large clashes between some atoms, which can cause explosion of the target molecule, and push some atoms out of the density map. The problems to be solved are almost same with those in the SHAKE errors (see above).

Compute_Energy_Restraints_Pos> Positional restraint energy is too big

This message indicates that some atoms to be restrained are significantly deviated from the reference position, indicating that the restraint might not be properly applied to such atoms. This situation is not allowed in **SPDYN**.

- Use a larger force constant to keep their position near the reference.
- Turn off the positional restraint for such atoms if it is not essential.

CHAPTER TWENTYTHREE

APPENDIX

23.1 Install the requirements in Linux

In the first sub-section, we explain how to install the requirements using the package manager “apt” in Ubuntu/Debian. If you want to install them from the source codes, or if you want to use other Linux systems like CentOS, please see the second sub-section.

23.1.1 For Ubuntu/Debian users

GNU compilers and build tools

First, we install compilers and build tools.

```
$ sudo apt update
$ sudo apt install build-essential
$ sudo apt install gfortran autoconf automake
```

OpenMPI

Then, we install OpenMPI. Note that the development version (XXX-dev) should be installed.

```
$ sudo apt install openmpi-bin libopenmpi-dev

$ which mpirun mpif90 mpicc
/usr/bin/mpirun
/usr/bin/mpif90
/usr/bin/mpicc
```

LAPACK/BLAS libraries

Finally, we install LAPACK/BLAS libraries. Again, development version (XXX-dev) is installed.

```
$ sudo apt install liblapack-dev

$ ls /usr/lib/x86_64-linux-gnu/liblapack.*
$ ls /usr/lib/x86_64-linux-gnu/libblas.*
```

23.1.2 For CentOS/RedHat users

Here, we explain how to install OpenMPI and LAPACK/BLAS libraries from the source codes. We assume that the users already installed GNU compilers. The following schemes are commonly applicable to typical Linux systems including CentOS and Red Hat.

OpenMPI

The source code of OpenMPI is availabe in https://www.open-mpi.org/. The following commands install OpenMPI 3.1.5 in the user's local directory "$HOME/Software/mpi" as an example. Here, we use GNU compilers (gcc, g++, and gfortran).

```
$ cd $HOME
$ mkdir Software
$ cd Software

$ mkdir build
$ cd build

$ wget https://download.open-mpi.org/release/open-mpi/v3.1/openmpi-3.1.5.
↪tar.gz

$ tar -xvf openmpi-3.1.5.tar.gz
$ cd openmpi-3.1.5

$ ./configure --prefix=$HOME/Software/mpi CC=gcc CXX=g++ F77=gfortran␣
↪FC=gfortran

$ make all
$ make install
```

The following information is added in "~/.bash_profile" (or "~/.bashrc").

```
MPIROOT=$HOME/Software/mpi
export PATH=$MPIROOT/bin:$PATH
export LD_LIBRARY_PATH=$MPIROOT/lib:$LD_LIBRARY_PATH
export MANPATH=$MPIROOT/share/man:$MANPATH
```

Launch another terminal window or reload "~/.bash_profile" (or "~/.bashrc"):

```
$ source ~/.bash_profile
```

The OpenMPI tools should be installed in "$HOME/Software/mpi/bin".

```
$ which mpirun mpif90 mpicc
~/Software/mpi/bin/mpirun
~/Software/mpi/bin/mpif90
~/Software/mpi/bin/mpicc
```

If you want to uninstall OpenMPI, just remove the directory "mpi" in "Software".

LAPACK/BLAS libraries

The source code of LAPACK/BLAS is availabe in http://www.netlib.org/lapack/. The following commands install LAPACK 3.8.0 in the user's local directory "$HOME/Software/lapack-3.8.0" as an example. The BLAS library is also installed. We use GNU compilers (gcc and gfortran).

```
$ cd $HOME/Software
$ wget http://www.netlib.org/lapack/lapack-3.8.0.tar.gz
$ tar -xvf lapack-3.8.0.tar.gz
$ cd lapack-3.8.0

$ cp make.inc.example make.inc
$ make blaslib
$ make lapacklib

$ ls lib*
liblapack.a  librefblas.a

$ ln -s librefblas.a ./libblas.a
```

The following information is added in "~/.bash_profile" (or "~/.bashrc").

```
export LAPACK_PATH=$HOME/Software/lapack-3.8.0
```

Launch another terminal window or reload "~/.bash_profile" (or "~/.bashrc"):

```
$ source ~/.bash_profile
```

If you want to uninstall LAPACK/BLAS, just remove the directory "lapack-3.8.0" in "Software".

23.2 Install the requirements in Mac

We recommend the Mac users to utilize “Xcode” for the installation of GENESIS, and also to install “OpenMPI” from the source code to avoid a “clang” problem (see below).

23.2.1 Install general tools

Xcode and Homebrew

“Xcode” is available in the Mac App Store (https://developer.apple.com/xcode/), and it is free of charge. After the installation of Xcode, all tasks described below will be done on “Terminal”. The “Terminal app” is in the “Utilities” folder in Applications. Please launch the Terminal. This terminal is almost same with that in Linux.

We recommend you to further install “Homebrew”, which enables easy installation of various tools such as compilers. If you have already installed “MacPorts”, you do not need to install “Homebrew” to avoid a conflict between “Homebrew” and “MacPorts”. In the Homebrew website (https://brew.sh/), you can find a long command like “`/usr/bin/ruby -e "$(curl -fsSL https://...`”. To install homebrew, execute that command in the Terminal prompt.

GNU compilers and build tools

First, we install “gcc”, “autoconf”, “automake”, and other tools via homebrew:

```
$ brew install gcc
$ brew install autoconf
$ brew install automake
$ brew install wget
```

To confirm the installation of “gcc”, let us type the following commands:

```
$ which gcc
/usr/bin/gcc

$ gcc --version
...
Apple LLVM version 10.0.1 (clang-1001.0.46.4)
```

These messages tell us that “gcc” is installed in the “/usr/bin” directory. However, this gcc is not a “real” GNU compiler, and it is linked to another compiler “clang”. If you use this gcc for the installation of OpenMPI, it can cause a trouble in compiling **GENESIS** with a certain option. Therefore, you have to use a “real” GNU compiler, which is actually installed in “/usr/local/bin”. For example, if you have installed gcc ver. 9, you can find it as “gcc-9” in “/usr/local/bin”.

```
$ ls /usr/local/bin/gcc*
/usr/local/bin/gcc-9      /usr/local/bin/gcc-ar-9      ...

$ gcc-9 --version
gcc-9 (Homebrew GCC 9.2.0) 9.2.0
```

23.2.2 Install libraries

OpenMPI

We then install "OpenMPI". We specify "real" GNU compilers explicitly in the configure command. The following commands install OpenMPI in the user's local directory "$HOME/Software/mpi".

```
$ cd $HOME
$ mkdir Software
$ cd Software
$ mkdir build
$ cd build

$ wget https://download.open-mpi.org/release/open-mpi/v3.1/openmpi-3.1.5.
↪tar.gz

$ tar -xvf openmpi-3.1.5.tar.gz
$ cd openmpi-3.1.5

$ ./configure --prefix=$HOME/Software/mpi CC=gcc-9 CXX=g++-9 F77=gfortran-
↪9 FC=gfortran-9

$ make all
$ make install
```

The following information is added in "~/.bash_profile" (or "~/.bashrc").

```
MPIROOT=$HOME/Software/mpi
export PATH=$MPIROOT/bin:$PATH
export LD_LIBRARY_PATH=$MPIROOT/lib:$LD_LIBRARY_PATH
export MANPATH=$MPIROOT/share/man:$MANPATH
```

Launch another terminal window or reload "~/.bash_profile" (or "~/.bashrc"):

```
$ source ~/.bash_profile
```

The OpenMPI tools should be installed in "$HOME/Software/mpi/bin".

```
$ which mpirun mpif90 mpicc
/Users/[username]/Software/mpi/bin/mpirun
/Users/[username]/Software/mpi/bin/mpif90
/Users/[username]/Software/mpi/bin/mpicc
```

Make sure that "mpicc" and "mpif90" are linked to "gcc-9" and "gfortran-9", respectively.

```
$ mpicc --version
gcc-9 (Homebrew GCC 9.2.0) 9.2.0

$ mpif90 --version
GNU Fortran (Homebrew GCC 9.2.0) 9.2.0
```

If you want to uninstall OpenMPI, just remove the directory "mpi" in "Software".

LAPACK/BLAS libraries

Finally, we install LAPACK and BLAS libraries. Again, "real" GNU compilers are used for the install.

```
$ cd $HOME/Software
$ wget http://www.netlib.org/lapack/lapack-3.8.0.tar.gz
$ tar -xvf lapack-3.8.0.tar.gz
$ cd lapack-3.8.0

$ cp make.inc.example make.inc
```

In the "make.inc" file, there are three lines to be modified:

```
#  CC is the C compiler, normally invoked with options CFLAGS.
#
CC     = gcc-9
...

#  should not compile LAPACK with flags such as -ffpe-trap=overflow.
#
FORTRAN = gfortran-9
...

#  load options for your machine.
#
LOADER   = gfortran-9
...
```

After the modification, we install BLAS and LAPACK libraries:

```
$ make blaslib
$ make lapacklib

$ ls lib*
liblapack.a  librefblas.a

$ ln -s librefblas.a ./libblas.a
```

The following information is added in "~/.bash_profile" (or "~/.bashrc").

```
export LAPACK_PATH=$HOME/Software/lapack-3.8.0
```

Launch another terminal window or reload "~/.bash_profile" (or "~/.bashrc"):

```
$ source ~/.bash_profile
```

If you want to uninstall LAPACK/BLAS, just remove the directory "lapack-3.8.0" in "Software".

BIBLIOGRAPHY

[1] D.A. Case, I.Y. Ben-Shalom, S.R. Brozell, D.S. Cerutti, T.E. Cheatham III, V.W.D. Cruzeiro, T.A. Darden, R.E. Duke, D. Ghoreishi, M.K. Gilson, H. Gohlke, A.W. Goetz, D. Greene, R Harris, N. Homeyer, S. Izadi, A. Kovalenko, T. Kurtzman, T.S. Lee, S. LeGrand, P. Li, C. Lin, J. Liu, T. Luchko, R. Luo, D.J. Mermelstein, K.M. Merz, Y. Miao, G. Monard, C. Nguyen, H. Nguyen, I. Omelyan, A. Onufriev, F. Pan, R. Qi, D.R. Roe, A. Roitberg, C. Sagui, S. Schott-Verdugo, J. Shen, C.L. Simmerling, J. Smith, R. Salomon-Ferrer, J. Swails, R.C. Walker, J. Wang, H. Wei, R.M. Wolf, X. Wu, L. Xiao, D.M. York, and P.A. Kollman. Amber18. University of California, San Francisco, 2018.

[2] B. R. Brooks, C. L. Brooks, A. D. Mackerell, L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, and M. Karplus. CHARMM: The biomolecular simulation program. *J. Comput. Chem.*, 30:1545–1614, 2009. URL: http://dx.doi.org/10.1002/jcc.21287, doi:10.1002/jcc.21287 (https://doi.org/10.1002/jcc.21287).

[3] S. Pronk, S. Páll, R. Schulz, P. Larsson, P. Bjelkmar, R. Apostolov, M. R. Shirts, J. C. Smith, P. M. Kasson, D. van der Spoel, B. Hess, and E. Lindahl. GROMACS 4.5: a high-throughput and highly parallel open source molecular simulation toolkit. *Bioinformatics*, 29:845–854, 2013.

[4] K. J. Bowers, E. Chow, H. Xu, R. O. Dror, M. P. Eastwood, B. A. Gregersen, J. L. Klepeis, I. Kolossvary, M. A. Moraes, F. D. Sacerdoti, J. K. Salmon, Y. Shan, and D. E. Shaw. Scalable algorithms for molecular dynamics simulations on commodity clusters. In *SC 2006 Conference, Proceedings of the ACM/IEEE*, 11–17. IEEE, 2006.

[5] J. C. Phillips, R. Braun, W. Wang, J. Gumbart, E. Tajkhorshid, E. Villa, C. Chipot, R. D. Skeel, L. Kalé, and K. Schulten. Scalable molecular dynamics with NAMD. *J. Comput. Chem.*, 26:1781–1802, 2005. URL: http://dx.doi.org/10.1002/jcc.20289, doi:10.1002/jcc.20289 (https://doi.org/10.1002/jcc.20289).

[6] D. A. Case, T. E. Cheatham, T. Darden, H. Gohlke, R. Luo, K. M. Merz, A. Onufriev, C. Simmerling, B. Wang, and R. J. Woods. The Amber biomolecular simulation programs. *J. Comput. Chem.*, 26:1668–1688, 2005.

[7] A. D. MacKerell, D. Bashford, M. Bellott, R. L. Dunbrack, J. D. Evanseck, M. J. Field, S. Fischer, J. Gao, H. Guo, S. Ha, D. Joseph-McCarthy, L. Kuchnir, K. Kuczera, F. T. K. Lau, C. Mattos, S. Michnick, T. Ngo, D. T. Nguyen, B. Prodhom, W. E. Reiher, B. Roux, M. Schlenkrich, J. C. Smith, R. Stote, J. Straub, M. Watanabe, J. Wiorkiewicz-Kuczera, D. Yin, and M. Karplus. All-atom empirical potential for molecular modeling and dynamics studies of proteins. *J. Phys. Chem. B*, 102:3586–3616, 1998.

[8] A. D. MacKerell, M. Feig, and C. L. Brooks. Improved treatment of the protein backbone in empirical force fields. *J. Am. Chem. Soc.*, 126:698–699, 2004.

[9] W. L. Jorgensen, D. S. Maxwell, and J. Tirado-Rives. Development and Testing of the OPLS All-Atom Force Field on Conformational Energetics and Properties of Organic Liquids. *J. Am. Chem. Soc.*, 118:11225–11236, 1996.

[10] C. Oostenbrink, A. Villa, A. E. Mark, and W. F. Van Gunsteren. A biomolecular force field based on the free enthalpy of hydration and solvation: The GROMOS force-field parameter sets 53A5 and 53A6. *J. Comput. Chem.*, 25:1656–1676, 2004.

[11] J. Jung, T. Mori, and Y. Sugita. Efficient lookup table using a linear function of inverse distance squared. *J. Comput. Chem.*, 34:2412–2420, 2013.

[12] J. Jung, T. Mori, and Y. Sugita. Midpoint cell method for hybrid (MPI+OpenMP) parallelization of molecular dynamics simulations. *J. Comput. Chem.*, 35:1064–1072, 2014.

[13] J. Huang, S. Rauscher, G. Nawrocki, T. Ran, M. Feig, B. L. de Groot, H. Grubmüller, and A. D. MacKerell Jr. CHARMM36m: an improved force field for folded and intrinsically disordered proteins. *Nat. Methods*, 14:71–73, 2017.

[14] W. Humphrey, A. Dalke, and K. Schulten. VMD: Visual molecular dynamics. *J. Mol. Graph.*, 14:33–38, 1996.

[15] S. Jo, T. Kim, V. G. Iyer, and W. Im. CHARMM GUI: A web based graphical user interface for CHARMM. *J. Comput. Chem.*, 29:1859–1865, 2008.

[16] W. D. Cornell, P. Cieplak, C. I. Bayly, I. R. Gould, K. M. Merz, D. M. Ferguson, D. C. Spellmeyer, T. Fox, J. W. Caldwell, and P. A. Kollman. A Second Generation Force Field for the Simulation of Proteins, Nucleic Acids, and Organic Molecules. *J. Am. Chem. Soc.*, 117:5179–5197, 1995.

[17] H. Taketomi, Y. Ueda, and N. Go. Studies on protein folding, unfolding and fluctuations by computer simulation. *nt. J. Peptide Proteins Res.*, 7:445–459, 1975.

[18] S.J. Marrink, H.J. Risselada, S. Yefimov, D.P. Tieleman, and A.H. de Vries. The MARTINI forcefield: coarse grained model for biomolecular simulations. *J. Phys. Chem. B*, 111:7812–7824, 2007.

[19] P. C. Whitford, J. K. Noel, S. Gosavi, A. Schug, K. Y. Sanbonmatsu, and J. N. Onuchic. An all-atom structure-based potential for proteins: Bridging minimal models with all-atom empirical forcefields. *Proteins: Structure, Function, and Bioinformatics*, 75:430–441, 2009.

[20] J. K. Noel, P. C. Whitford, K. Y. Sanbonmatsu, and J. N. Onuchic. SMOG@ctbp: simplified deployment of structure based models in GROMACS. *Nucleic Acids Res.*, 38:W657–W661, 2010.

[21] J. K. Noel, M. Levi, M. Raghunathan, H. Lammert, R. L. Hayes, J. N. Onuchic, and P. C. Whitford. SMOG 2: A Versatile Software Package for Generating Structure Based Models. *PLoS Comput. Biol.*, 12:e1004794, 2016.

[22] J. Karanicolas and C. L. Brooks, III. The origins of asymmetry in the folding transition states of protein L and protein G. *Protein Sci.*, 11:2351–2361, 2002.

[23] J. Karanicolas and C. L. Brooks III. Improved Gō-like models demonstrate the robustness of protein folding mechanisms towards non-native interactions. *J. Mol. Biol.*, 334:309–325, 2003.

[24] M. Feig, J. Karanicolas, and C. L. III Brooks. MMTSB Tool Set: enhanced sampling and multiscale modeling methods for applications in structural biology. *J. Mol. Graph, Model.*, 22:377–395, 2004.

[25] CHARMM. http://www.charmm.org/.

[26] AMBER. http://ambermd.org/.

[27] Gromacs. http://www.gromacs.org/.

[28] J. B. Klauda, R. M. Venable, J. A. Freites, J. W. O'Connor, D. J. Tobias, C. Mondragon-Ramirez, I. Vorobyov, and R. W. Pastor. Update of the charmm all-atom additive force field for lipids: validation on six lipid types. *J. Phys. Chem. B*, 114:7830–7843, 2010.

[29] R. B. Best, X. Zhu, J. Shim, P. E. M. Lopes, J. Mittal, M. Feig, and A. D. MacKerell. Optimization of the additive CHARMM all-atom protein force field targeting improved sampling of the backbone \phi , \psi and side-Chain \chi 1 and \chi 2 dihedral angles. *J. Chem. Theo. Comput.*, 8:3257–3273, 2012.

[30] J. Huang and A. D. MacKerell. CHARMM36 all-atom additive protein force field: Validation based on comparison to NMR data. *J. Comput. Chem.*, 34:2135–2145, 2013.

[31] L. Monticelli, S.K. Kandasamy, X. Periole, R.G. Larson, D.P. Tieleman, and S.J. Marrink. The MARTINI coarse grained forcefield: extension to proteins. *J. Chem. Theo. Comput.*, 4:819–834, 2008.

[32] C. Clementi, H. Nymeyer, and J. Onuchic. Topological and energetic factors: what determines the structural details of the transition state ensemble and "en-route" intermediates for protein folding? an investigation for small globular proteins. *J. Mol. Biol.*, 298:937–953, 2000.

[33] L. Verlet. Computer Experiments on Classical Fluids .I. Thermodynamical Properties of Lennard-Jones Molecules. *Phys. Rev.*, 159:98–103, 1967.

[34] P. J. Steinbach and B. R. Brooks. New Spherical-Cutoff Methods for Long-Range Forces in Macromolecular Simulation. *J. Comput. Chem.*, 15:667–683, 1994.

[35] T. Darden, D. York, and L. Pedersen. Particle mesh Ewald: An Nlog(N) method for Ewald sums in large systems. *J. Chem. Phys.*, 98:10089–10092, 1993.

[36] U. Essmann, L. Perera, M. L. Berkowitz, T. Darden, H. Lee, and L. G. Pedersen. A smooth particle mesh Ewald method. *J. Chem. Phys.*, 103:8577–8593, 1995.

[37] J. Jung, C. Kobayashi, T. Imamura, and Y. Sugita. Parallel implementation of 3D FFT with volumetric decomposition schemes for efficient molecular dynamics simulations. *Comp. Phys. Comm.*, 200:57–65, 2016.

[38] D. Takahashi. FFTE: A Fast Fourier Transform Package. http://www.ffte.jp/.

[39] L. Nilsson. Efficient Table Lookup Without Inverse Square Roots for Calculation of Pair Wise Atomic Interactions in Classical Simulations. *J. Comput. Chem.*, 30:1490–1498, 2009.

[40] T. Mori, N. Miyashita, W. Im, M. Feig, and Y. Sugita. Molecular dynamics simulations of biological membranes and membrane proteins using enhanced conformational sampling algorithms. *BBA-Biomembranes*, 1858:1635–1651, 2016.

[41] W. C. Still, A. Tempczyk, R. C. Hawley, and T. Hendrickson. Semianalytical treatment of solvation for molecular mechanics and dynamics. *J. Am. Chem. Soc.*, 112:6127–6129, 1990.

[42] D. Eisenberg and A. D. McLachlan. Solvation energy in protein folding and binding. *Nature*, 319:199–203, 1986.

[43] A. Onufriev, D. Bashford, and D. A. Case. Exploring protein native states and large scale conformational changes with a modified generalized born model. *Proteins*, 55:383–394, 2004.

[44] J. Weiser, P. S. Shenkin, and W. C. Still. Approximate atomic surfaces from linear combinations of pairwise overlaps (LCPO). *J. Comput. Chem.*, 20:217–230, 1999.

[45] M. Schaefer and C. Froemmel. A Precise Analytical Method for Calculating the Electrostatic Energy of Macromolecules in Aqueous Solution. *J. Mol. Biol.*, 216:1045–1066, 1990.

[46] M. Tuckerman, B. J. Berne, and Martyna G. J. Reversible multiple time scale molecular dynamics. *J. Chem. Phys.*, 97:1990–2001, 1992.

[47] J. Schlitter, M. Engels, and P. Kruger. Targeted molecular dynamics: a new approach for searching pathways of conformational transitions. *J. Mol. Graph.*, 12:84–89, 1994.

[48] J. P. Ryckaert, G. Ciccotti, and H. J. C. Berendsen. Numerical-Integration of Cartesian Equations of Motion of a System with Constraints - Molecular-Dynamics of N-Alkanes. *J. Comput. Chem.*, 23:327–341, 1977.

[49] H. C. Andersen. Rattle - a Velocity Version of the Shake Algorithm for Molecular-Dynamics Calculations. *J. Comput. Chem.*, 52:24–34, 1983.

[50] S. Miyamoto and P. A. Kollman. Settle - an Analytical Version of the Shake and Rattle Algorithm for Rigid Water Models. *J. Comput. Chem.*, 13:952–962, 1992.

[51] S. A. Adelman and J. D. Doll. Generalized Langevin Equation Approach for Atom-Solid-Surface Scattering - General Formulation for Classical Scattering Off Harmonic Solids. *J. Chem. Phys.*, 64:2375–2388, 1976.

[52] D. Quigley and M. I. J. Probert. Langevin dynamics in constant pressure extended systems. *J. Chem. Phys.*, 120:11432–11441, 2004.

[53] Y. Zhang, S. E. Feller, B. R. Brooks, and Pastor R. W. Computer simulation of liquid/liquid interfaces. I. Theory and application to octane/water. *J. Chem. Phys.*, 103:10252–10266, 1995.

[54] H. J. C. Berendsen, J. P. M. Postma, W. F. Vangunsteren, A. Dinola, and J. R. Haak. Molecular-Dynamics with Coupling to an External Bath. *J. Chem. Phys.*, 81:3684–3690, 1984.

[55] G. Bussi, D. Donadio, and M. Parrinello. Canonical sampling through velocity rescaling. *J. Chem. Phys.*, 126:014101, 2007.

[56] G. Bussi, T. Zykova-Timan, and M. Parrinello. Isothermal-isobaric molecular dynamics using stochastic velocity rescaling. *J. Chem. Phys.*, 130:074101, 2009.

[57] C. Kandt, W. L. Ash, and D. P. Tieleman. Setting up and running molecular dynamics simulations of membrane proteins. *Method*, 41:475–488, 2007.

[58] Y. Sugita and Y. Okamoto. Replica-exchange molecular dynamics method for protein folding. *Chem. Phys. Lett.*, 314:141–151, 1999.

[59] A. Mitsutake, Y. Sugita, and Y. Okamoto. Generalized-ensemble algorithms for molecular simulations of biopolymers. *Biopolymers*, 60:96–123, 2001.

[60] Y. Mori and Y. Okamoto. Generalized-ensemble algorithms for the isobaric-isothermal ensemble. *J. Phys. Soc. Jpn.*, 79:074003, 2010.

[61] Y. Mori and Y. Okamoto. Replica-exchange molecular dynamics simulations for various constant temperature algorithms. *J. Phys. Soc. Jpn.*, 79:074001, 2010.

[62] T. Okabe, M. Kawata, Y. Okamoto, and M. Mikami. Replica-exchange Monte Carlo method for the isobaric-isothermal ensemble. *Chem. Phys. Lett.*, 335:435–439, 2001.

[63] T. Mori, J. Jung, and Y. Sugita. Surface-tension replica-exchange molecular dynamics method for enhanced sampling of biological membrane systems. *J. Chem. Theory. Comput.*, 9:5629–5640, 2013.

[64] Y. Sugita, A. Kitao, and Y. Okamoto. Multidimensional replica-exchange method for free-energy calculations. *J. Chem. Phys.*, 113:6042–6051, 2000.

[65] H. Fukunishi, O. Watanabe, and S. Takada. On the Hamiltonian replica exchange method for efficient sampling of biomolecular systems: Application to protein structure prediction. *J. Chem. Phys.*, 116:9058–9067, 2002.

[66] T. Terakawa, T. Kameda, and S. Takada. On Easy Implementation of a Variant of the Replica Exchange with Solute Tempering in GROMACS. *J. Comput. Chem.*, 32:1228–1234, 2011.

[67] M. Kamiya and Y. Sugita. Flexible selection of the solute region in replica exchange with solute tempering: Application to protein-folding simulations. *J. Chem. Phys.*, 149:072304, 2018.

[68] P. Liu, B. Kim, R. A. Friesner, and B. J. Berne. Replica exchange with solute tempering: A method for sampling biological systems in explicit water. *Proc. Natl. Acad. Sci. USA*, 102:13749–13754, 2005.

[69] W. Jiang, M. Hodoscek, and B. Roux. Computation of Absolute Hydration and Binding Free Energy with Free Energy Perturbation Distributed Replica-Exchange Molecular Dynamics. *J. Chem. Theory Comput.*, 5:2583–2588, 2009.

[70] S. Re, H. Oshima, K. Kasahara, M. Kamiya, and Y. Sugita. Encounter complexes and hidden poses of kinase-inhibitor binding on the free-energy landscape. *Proc. Natl. Acad. Sci. USA*, 116:18404–18409, 2019.

[71] L Maragliano, A. Fischer, E. Vanden-Eijnden, and G. Ciccotti. String method in collective variables: minimum free energy paths and isocommittor surfaces. *J. Chem. Phys.*, 125:24106, 2006.

[72] L. Maragliano and E. Vanden-Eijnden. On-the-fly string method for minimum free energy paths calculation. *Chem. Phys. Lett.*, 446:182–190, 2007.

[73] A. C Pan, D. Sezer, and B. Roux. Finding transition pathways using the string method with swarms of trajectories. *J. Phys. Chem. B*, 112:3432–3440, 2008.

[74] Y. Matsunaga, Y. Komuro, C. Kobayashi, J. Jung, T. Mori, and Y. Sugita. Dimensionality of Collective Variables for Describing Conformational Changes of a Multi-Domain Protein. *J. Phys. Chem. Lett.*, 7:1446–1451, 2016.

[75] W. E, W. Ren, and E. Vanden-Eijnden. Simplified and improved string method for computing the minimum energy paths in barrier-crossing events. *J. Chem. Phys.*, 126:164103, 2007.

[76] D. Sheppard, R. Terrell, and G. Henkelman. Optimization methods for finding minimum energy paths. *J. Chem. Phys.*, 128:134106, 2008.

[77] Y. Miao, V. A. Feher, and J. A. McCammon. Gaussian Accelerated Molecular Dynamics: Unconstrained Enhanced Sampling and Free Energy Calculation. *J. Chem. Theory Comput.*, 11:3584–3595, 2015.

[78] Y. T. Pang, Y. Miao, Y. Wang, and J. A. McCammon. Gaussian Accelerated Molecular Dynamics in NAMD. *J. Chem. Theory Comput.*, 13:9–19, 2017.

[79] D. Hamelberg, J. Mongan, and J. A. McCammon. Accelerated Molecular Dynamics: A Promising and Efficient Simulation Method for Biomolecules. *J. Chem. Phys.*, 120:11919–11929, 2004.

[80] D. Hamelberg, C. A. F. de Oliveira, and J. A. McCammon. Sampling of Slow Diffusive Conformational Transitions with Accelerated Molecular Dynamics. *J. Chem. Phys.*, 127:155102, 2007.

[81] T. Shen and D. Hamelberg. A Statistical Analysis of the Precision of Reweighting-Based Simulations. *J. Chem. Phys.*, 129:034103, 2008.

[82] Y. Miao, W. Sinko, L. Pierce, D. Bucher, R. C. Walker, and J. A. McCammon. Improved Reweighting of Accelerated Molecular Dynamics Simulations for Free Energy Calculation. *J. Chem. Theory Comput.*, 10:2677–2689, 2014.

[83] H. Oshima, S. Re, and Y. Sugita. Replica-Exchange Umbrella Sampling Combined with Gaussian Accelerated Molecular Dynamics for Free-Energy Calculation of Biomolecules. *J. Chem. Theory Comput.*, 2019. URL: https://pubs.acs.org/doi/10.1021/acs.jctc.9b00761, doi:10.1021/acs.jctc.9b00761 (https://doi.org/10.1021/acs.jctc.9b00761).

[84] A. Warshel and M. Karplus. Calculation of Ground and Excited State Potential Surfaces of Conjugated Molecules. I. Formulation and Parametrization. *J. Am. Chem. Soc.*, 94:5612–5625, 1972.

[85] A. Warshel and M. Levitt. Theoretical studies of enzymic reactions: Dielectric, electrostatic and steric stabilization of the carbonium ion in the reaction of lysozyme. *J. Mol. Biol.*, 103:227–249, 1976.

[86] K. Yagi, K. Yamada, C. Kobayashi, and Y. Sugita. Anharmonic Vibrational Analysis of Biomolecules and Solvated Molecules Using Hybrid QM/MM Computations. *J. Chem. Theory Comput.*, 15:1924–1938, 2019.

[87] F. Tama, O. Miyashita, and C. L. Brooks. Flexible multi scale fitting of atomic structures into low resolution electron density maps with elastic network normal mode analysis. *J. Mol. Biol.*, 337:985–999, 2004.

[88] M. Orzechowski and F. Tama. Flexible fitting of high resolution X ray structures into cryoelectron microscopy maps using biased molecular dynamics simulations. *Biophys. J.*, 95:5692–5705, 2008.

[89] L. G. Trabuco, E. Villa, K. Mitra, J. Frank, and K. Schulten. Flexible fitting of atomic structures into electron microscopy maps using molecular dynamics. *Structure*, 16:673–683, 2008.

[90] M. Topf, K. Lasker, B. Webb, H. Wolfson, W. Chiu, and A. Sali. Protein structure fitting and refinement guided by cryo EM density. *Structure*, 16:295–307, 2008.

[91] H. Ishida and A. Matsumoto. Free energy landscape of reverse tRNA translocation through the ribosome analyzed by electron microscopy density maps and molecular dynamics simulations. *PloS one*, 9:e101951, 2014.

[92] O. Miyashita, C. Kobayashi, T. Mori, Y. Sugita, and F. Tama. Flexible fitting to cryo-EM density map using ensemble molecular dynamics simulations. *J. Comput. Chem.*, 38:1447–1461, 2017.

[93] P. C. Whitford, A. Ahmed, Y. Yu, Hennelly, S. P., F. Tama, Spahn, C. M., J. N. Onuchic, and K. Y. Sanbonmatsu. Excited states of ribosome translocation revealed through integrative molecular modeling. *Proc. Natl. Acad. Sci. U.S.A.*, 108:18943–18948, 2011.

[94] T. Mori, M. Kulik, O. Miyashita, J. Jung, F. Tama, and Y. Sugita. Acceleration of cryo-EM flexible fitting for large biomolecular systems by efficient space partitioning. *Structure*, 27:161–174.e3, 2019.

[95] J. Gao, K. Kuczera, B. Tidor, and M. Karplus. Hidden thermodynamics of mutant proteins: A molecular dynamics analysis. *Science*, 244:1069–1072, 1989.

[96] D. A. Pearlman. A comparison of alternative approaches to free energy calculations. *J. Phys. Chem.*, 98:1487–1493, 1994.

[97] P. H. Axelsen and D. Li. Improved convergence in dual-topology free energy calculations through use of harmonic restraints. *J. Comput. Chem.*, 19:1278–1283, 1998.

[98] W. Jiang, C. Chipot, and B. Roux. Computing relative binding 791 affinity of ligands to receptor: An effective hybrid single-dual-topology free-energy perturbation approach in NAMD. *J. Chem. Inf. Model.*, 59:3794–3802, 2019.

[99] M. Zacharias, T. P. Straatsma, and J. A. McCammon. Separation-shifted scaling, a new scaling method for Lennard-Jones interactions in thermodynamic integration. *J. Chem. Phys.*, 100:9025–9031, 1994.

[100] T. Steinbrecher, I. Joung, and D. A. Case. Soft-core potentials in thermodynamic integration: Comparing one- and two-step transformations. *J. Comput. Chem.*, 32:3253–3263, 2011.